2007

Calibration method of an analytical propagation delay model

Jeremy M. Buan
San Jose State University

CALIBRATION METHOD OF AN ANALYTICAL PROPAGATION DELAY MODEL

A Thesis

Presented to

The Faculty of the Department of Electrical Engineering

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Jeremy M. Buan

May 2007

APPROVED FOR THE DEPARTMENT OF ELECTRICAL ENGINEERING

Dr. David Parent

Dr. Tamara Schmitz

Dr. Lili He

APPROVED FOR THE UNIVERSITY

ABSTRACT

CALIBRATION METHOD OF AN ANALYTICAL PROPAGATION DELAY MODEL

by Jeremy M. Buan

This thesis presents a new method of calibrating an analytical propagation delay model of CMOS gates. This method requires only twenty simulations to calibrate the inverter at various input slew rates, and a total of four simulations to calibrate various other digital gates. These calibrations make the propagation delay model more accurate by taking into account input slew rate dependence and velocity saturation. The propagation delay model used is analytical and accurate, allowing a designer to make appropriate tradeoffs between propagation delay, size, and power of a design. It is compared with various models in the literature. The method is verified with three different process technologies with the inverter, NAND, and NOR gates under various loading and input conditions. The model is further verified with a sixty-four bit adder design. This method is not only useful in designing CMOS gates, but also in designing circuit architectures.

Acknowledgements

I would like to thank Dr. Parent for all of his patience, guidance, and wisdom throughout the whole thesis process from beginning to end. I would also like to thank my parents for their infinite support. You guys have made this whole thing a lot easier for me. Finally I would like to thank my girlfriend and friends for being patient with me. Thanks to all of you again.

Table of Contents

List of Tables

List of Figures

Introduction

When it comes to full-custom CMOS digital circuit design, three aspects that a designer must take into account are propagation delay, area consumption, and power.[†] One would like to decrease delay to get the highest performance possible, but at the same time, to decrease the delay would mean to increase the area consumption and power. The larger the area consumption, the more expensive the circuit will be to manufacture and less features can be added. The higher the power consumption, the more heat extraction techniques will be required. Therefore, a circuit designer cannot simply make the transistor sizes arbitrarily large just to meet timing specifications.

Designers start off the design using simple, intuitive models to estimate transistor sizes. These initial estimates are then simulated in Simulation Program with Integrated Circuit Emphasis, also known as SPICE as in [1]. The sizes are then adjusted until timing specifications are met while ensuring that no more area is consumed than is necessary. If the models used during hand calculations are simple but inaccurate, a designer may waste an excessive amount of time readjusting the design in the simulation environment. On the other hand, if the models used for hand calculations are accurate but complex, one may not understand the tradeoffs of the model. A calibrated analytical model can be used to explore different architectures, but calibrating a model can also be time consuming.

A new method of calibration is presented in this thesis. This calibration method requires no more than twenty-four simulations. Approximately twenty simulations are

[†] Yield is important as well, but will not be discussed in this work.

needed to calibrate the standard inverter at various input slew rates. Only four more simulations are needed to calibrate other various gates such as NAND and NOR gates. This calibration method makes the design of digital gates more accurate because it considers input slew rate and carrier velocity saturation. A literature review is done before the method is presented. The method is then tested and verified with three different process technologies: AMI 0.6μm, TSMC 0.18μm, and IBM 0.13μm. The accuracy of this method is shown in the results. There are many cases in which this method is accurate to within twenty percent error, but there are also cases in which the error can reach up to sixty percent. This method is then compared to a widely used propagation delay model used to design digital gates called Logical Effort. The results of the verification of the method are analyzed. A 64-bit Kogg-Stone Adder is designed to show the application of the method to large-scale circuits. It is shown that the method can be used to estimate the propagation delay of a circuit consisting of many different digital gates.

Literature Review

To achieve the most optimal delay, area, or power, the full custom design technique is required, as stated in [50]. This is the best way to achieve high speed, area efficiency, and power efficiency because the width of each gate can be chosen. When designing digital gates using the full custom design flow, first a set of specifications for timing and loading are determined. Then an initial estimate for transistor dimensions is calculated. This initial design is simulated in SPICE to verify that timing specifications are met. If they are not met, transistor dimensions are recalculated, and the design is again simulated. This is sometimes referred to as a "simulate and tweak" cycle as stated in [2]. When the timing specifications are met, the circuit is properly sized. This process is illustrated in Figure 1.

When designing digital gates using the full custom design flow, it is important to estimate transistor dimensions before SPICE verification. To do this, a propagation delay model is needed. It is preferable to have a model that is simple and analytical, yet accurate, even though these are opposing criteria. The analytical model can explore the tradeoffs of propagation delay and power. The accuracy of the model will ensure that the designer does not spend too much time in the simulate-and-tweak cycle.

These desired traits, simplicity and accuracy, conflict with one another. There are models in the literature that are analytical and can be intuitively understood. The most accurate models of transistors to date, SPICE models, are not analytical and require a computer to calculate. An analytical equation will yield large discrepancies when

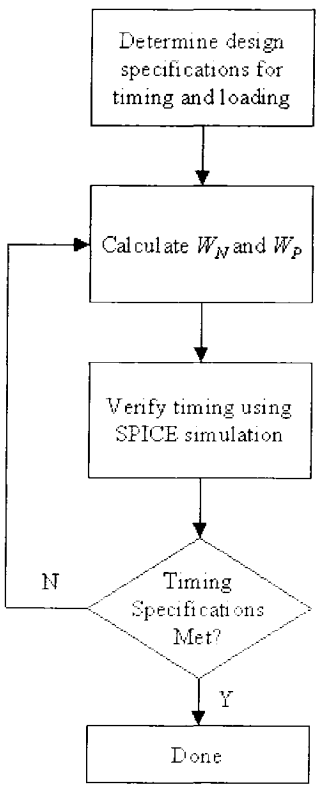

Figure 1. Full custom design flow for sizing digital gates.

compared to SPICE simulations. On the other hand, there are models in the literature that better fit the SPICE simulation results, [37]-[48]. These more accurate models are normally very complex and not very intuitive.

There are models in the literature that are both simple and accurate, [2] and [33]-[36]. This is achieved by calibrating the equations to fit SPICE simulation data. These calibrations require the collection of large amounts of SPICE simulation data.

This chapter provides some background information needed to understand the propagation delay models in the literature. The models in the literature will then be

reviewed. Three models worth exploring in detail are lookup tables, [33]; Logical Effort, [2]; and the Kang and Leblebici model, [3]. The tradeoffs of these models are included.

CMOS Gates

The most fundamental CMOS gate is the inverter as described in [3], shown in Figure 2. The inverter consists of one NMOS transistor and one PMOS transistor. The gates of these transistors are wired together to form the input. The drains of the transistors are wired together to form the output. The source of the PMOS transistor is tied to the power supply, V_{DD}. The source of the NMOS is tied to ground. When the input voltage is high, V_{DD}, the NMOS transistor is turned on while the PMOS transistor is turned off. This creates a path from the output to ground making the output voltage low. When the input voltage is low, the PMOS transistor is on while the NMOS transistor is off. This creates a path from the output to V_{DD} making the output voltage high.

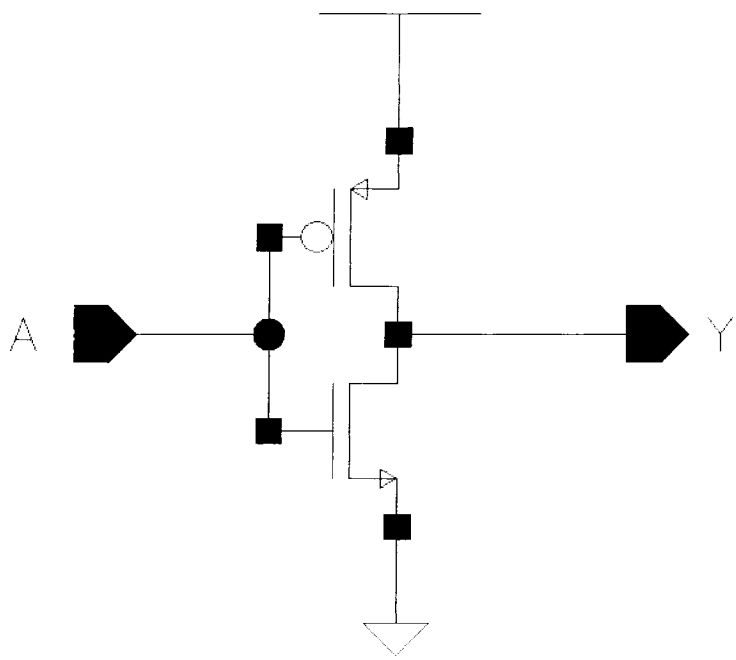

Figure 2. Schematic of inverter.

Other gates of practical use in digital circuits are NAND gates and NOR gates as described in [3]. NAND gates consist of multiple NMOS transistors connected in series and multiple PMOS transistors connected in parallel. In order for the output to reach ground, all inputs must be high. In order for the output to reach V_{DD}, only one input needs to be low. A three-input NAND gate is shown in Figure 3. NAND gates will be referred to as NANDX throughout the rest of the thesis, where X denotes the number of inputs. For instance, NAND4 refers to a four-input NAND gate.

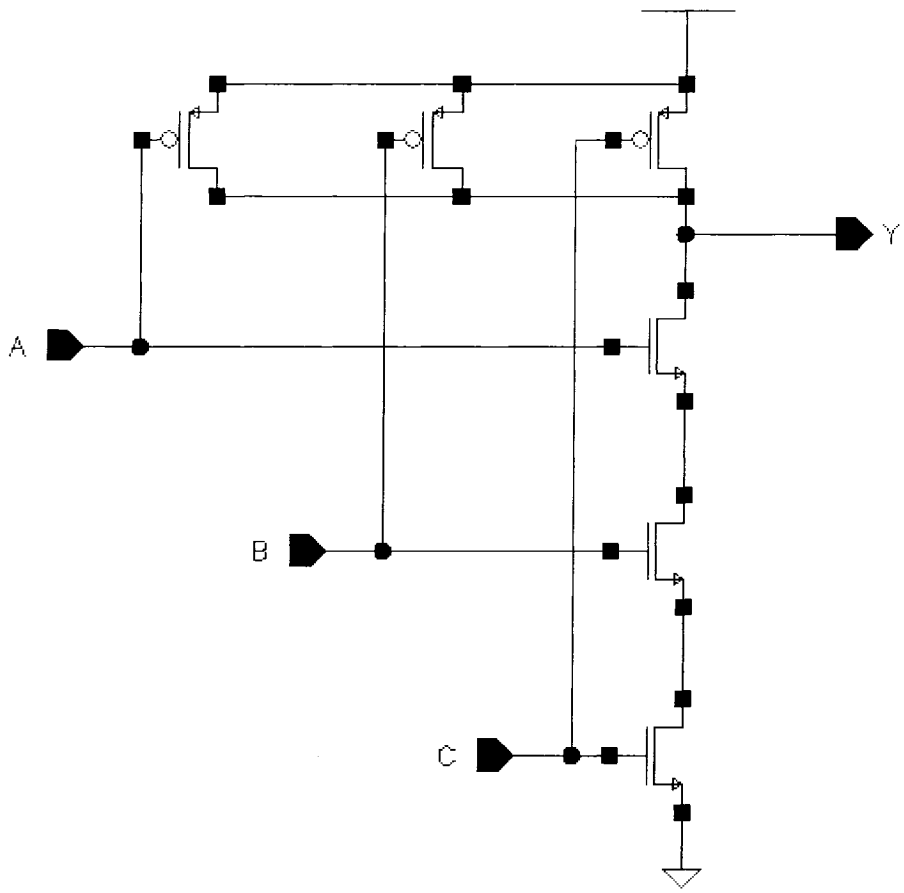

Figure 3. Schematic of a three-input NAND gate.

NOR gates consist of multiple PMOS transistors connected in series and multiple NMOS transistors connected in parallel. In order for the output to reach V_{DD}, all inputs

must be low. In order for the output to reach ground, only one input needs to be high. A three-input NOR gate is shown in Figure 4. NOR gates will be referred to as NORX throughout the rest of the thesis, where X denotes the number of inputs. For instance, NOR4 refers to a four-input NOR gate.

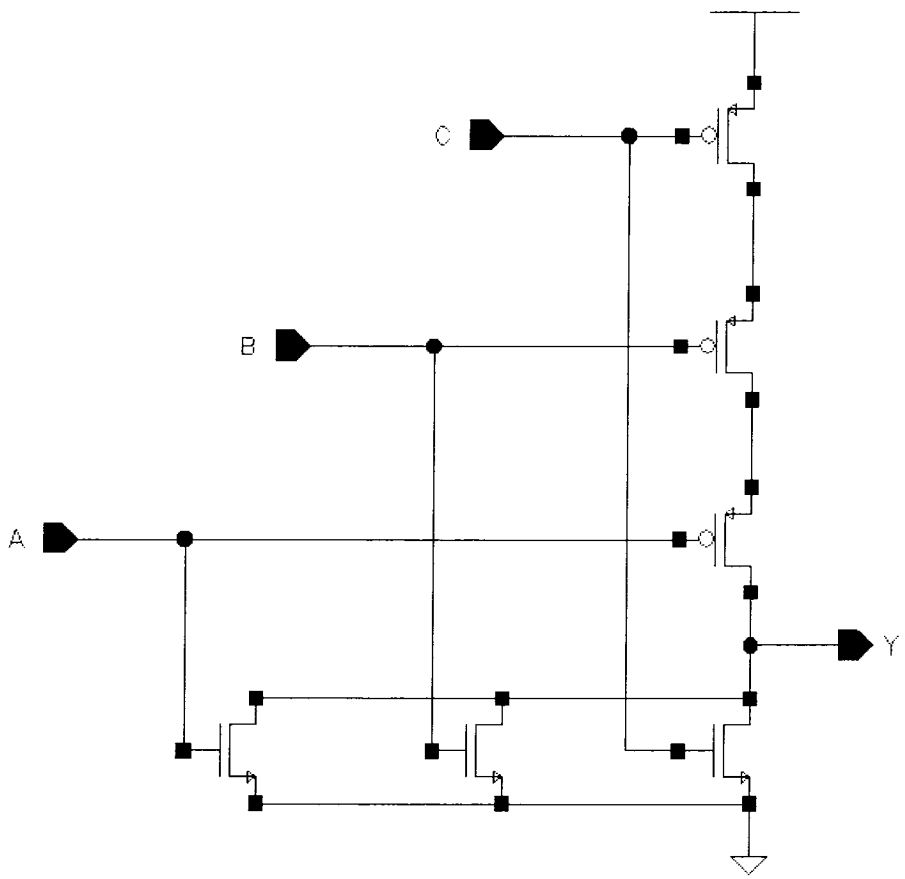

Figure 4. Schematic of a three-input NOR gate.

There are many other types of gates, but these are the three gates that this thesis is concentrating on: inverter, NAND and NOR gates.

Propagation Delay

When an input of a gate causes a change on the output, that change is not instantaneous. There is some delay between the change of the input and the change on

the output. A standard way to look at this delay is called the propagation delay. The propagation delay of a gate is the time difference between the time when the input reaches half of V_{DD} and the time when the output reaches half of V_{DD}. Half of V_{DD} is approximately the voltage on the input that triggers the change on the output. A graphical representation of propagation delay is shown in Figure 5. The delay of the output making a transition from high to low is represented as τ_{PHL}, and the delay of the output making a transition from low to high is represented as τ_{PLH}. In Figure 5, V_{DD} is 1.8V, and a line is drawn through half of V_{DD} to indicate the points used to calculate propagation delay.

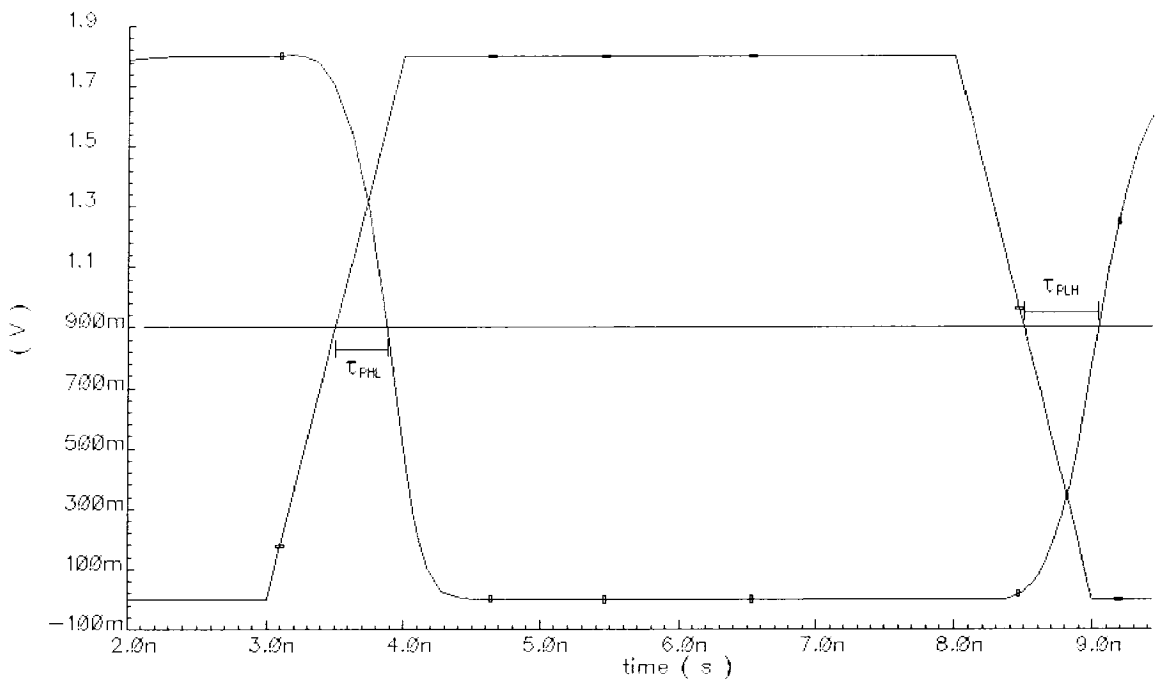

Figure 5. Input and output waveforms of an inverter showing propagation delays, τ_{PHL} and τ_{PLH}.

The propagation delay can be thought of as an RC delay, as mentioned in [3], with a resistance component and a capacitance component. The resistances in this case

are the resistances in the channels, sources, and drains of the transistors that are charging or discharging the capacitances. The capacitances in this case are the capacitances at the gates, sources, and drains of the transistors. Any propagation delay model reviewed throughout this thesis can be thought of as having a resistance component and a capacitance component.

Second Order Effects

Some effects that prevent the propagation delay from being a simple function are carrier velocity saturation, finite input slew rate, input-to-output coupling capacitance, body effect, threshold voltage dependence on drain voltage, and channel length modulation. These effects are explained in [3].

At low electric fields, the velocity of electrons is proportional to electric field. As the electric field increases beyond a certain magnitude, the velocity of electrons ceases to increase. The electrons reach a maximum velocity. This is what is known as velocity saturation as described in [3]. This saturation makes the dependence of current on V_{DS} more complex to model. The NMOS transistor of an inverter will definitely exhibit velocity saturation because all of the output voltage will be across the one NMOS transistor. The NMOS transistors of gates with multiple NMOS transistors in series may not exhibit velocity saturation because the output voltage is divided among multiple NMOS transistors. This difference makes it difficult to model these gates with the same delay equation.

To see how finite input slew rate and input-to-output coupling capacitance affect the propagation delay, look at Kirchoff's current law at the output of an inverter.

$$C_L \cdot \frac{dV_{OUT}}{dt} = C_{GD} \cdot \left(\frac{dV_{IN}}{dt} - \frac{dV_{OUT}}{dt}\right) + I_P - I_N \quad (1)$$

C_L is the capacitance from the output to ground. V_{OUT} is the voltage at the output node. C_{GD} is the capacitance between the gates of the transistors to the drains of the transistors. V_{IN} is the voltage at the input node. I_P is the current generated by the PMOS transistor, and I_N is the current generated by the NMOS transistor. This is shown graphically in Figure 6.

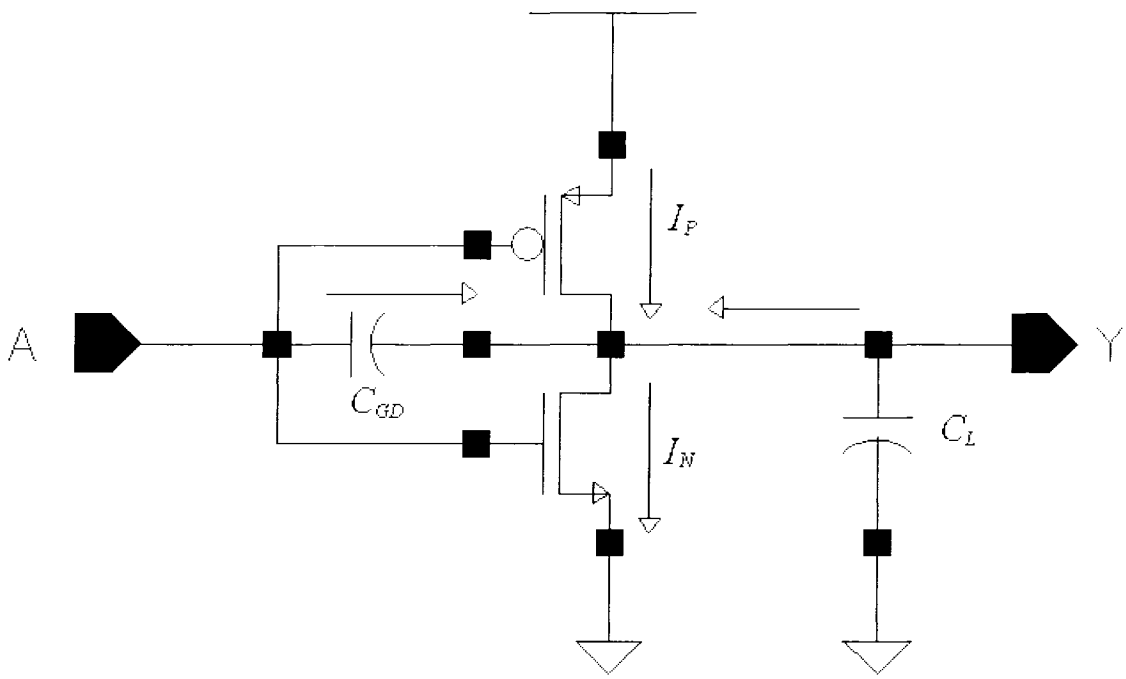

Figure 6. Kirchoff's current law on the output node of an inverter.

Equation (1) can be rearranged as in (2).

$$\frac{dV_{OUT}}{dt} = \frac{C_{GD}}{C_{GD} + C_L} \cdot \frac{dV_{IN}}{dt} + \frac{I_P - I_N}{C_{GD} + C_L} \quad (2)$$

By solving this differential equation, the propagation delay can be found. The solution to this differential equation is quite complex. Because V_{IN} does not change instantaneously,

I_P and I_N are not constant, and carrier mobility changes with terminal voltages. As V_{IN} rises from zero to V_{DD}, for instance, the NMOS transistor slowly turns on while the PMOS transistor slowly turns off. There are five different regions of operation that the inverter goes through during a transition. Equation (2) needs to be solved for each region. Even if the simple square law model were used for I_P and I_N, the solution would still be very complex. An example solution to (2) is in [5].

The square law model for an NMOS transistor is shown in (3) and (4).

$$I_N = \mu_N \cdot C_{OX} \cdot \frac{W}{L} \cdot [(V_{GS} - V_T) \cdot V_{DS} - \frac{1}{2} \cdot V_{DS}^2] \text{ (linear region)} \tag{3}$$

$$I_N = \mu_N \cdot C_{OX} \cdot \frac{W}{L} \cdot (V_{GS} - V_T)^2 \text{ (saturation region)} \tag{4}$$

μ_N is the mobility of the electrons in the channel of the NMOS transistor. C_{OX} is the gate oxide capacitance. W is the gate width. L is the gate length. V_{GS} is the gate to source voltage. V_T is the threshold voltage. V_{DS} is the drain to source voltage. These current equations are not accurate for short gate lengths. One reason for the decreased accuracy is velocity saturation as described above. The change in threshold voltage with body voltage and drain voltage variation also adds to discrepancies. Mobility also changes with V_{GS} and V_{DS}. Another reason for the inaccuracy of these equations is that it does not take into account channel length modulation in the saturation region. When in saturation region, this model shows that the current does not depend on V_{DS}. In reality, as V_{DS} increases so does current. These effects make the current model more complex, making the propagation delay more complex to model.

Propagation Delay Models in the Literature

There have been many attempts at modeling the propagation delay of CMOS gates in the literature, [2]-[48]. Some of these models are meant to model only the inverter, [4]-[25], and are not shown to be expandable to other gates. Some of these models are meant to model CMOS gates in general; [2], [3], and [26]-[48]. Some of these models, [3] and [26]-[32], have not been verified for process technologies with short channel effects. The accuracy of these models is questionable. Some of these models, [2] and [33]-[48], are very accurate. This high accuracy does have its price. Some of the models, [2] and [33]-[36], require excessive of calibration. Others, [37]-[48], cannot be put into a form that is convenient for circuit designers to calculate transistor sizes; they do not have a direct relationship between propagation delay and device geometries. In these case, they are good for characterization but not for design. Three models worth exploring in detail are lookup tables, Logical Effort, and the Kang and Leblebici model.

Lookup Tables

Some propagation delay models use a lookup table or mesh table as explained in [33]. Lookup tables can be used to design digital gates very quickly. They are simple and easy to use, but they are not very intuitive. By using a lookup table, a designer is not able to see the physical properties that govern the behavior of a circuit. Another drawback is that the amount of simulations needed to create the lookup tables can use excessive resources to create and maintain.

Lookup tables are arrays of simulated data under various parameters, such as load capacitance and input slew rate. A designer simply needs to take the design specifications and look for the results in the already made table. An example of a lookup table is shown in Figure 7.

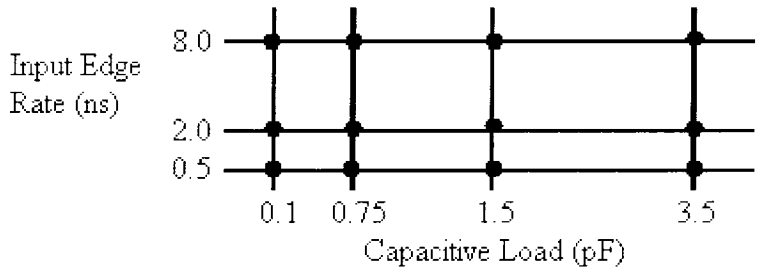

Figure 7. Example of a lookup table for propagation delay with respect to capacitive load and input edge rate.

Each point in the lookup table in Figure 7 represents a simulated propagation delay. Propagation delays between points are found using interpolation. This is a very accurate way of modeling propagation delay. A lookup table can also consist of device sizes needed for a given propagation delay and load capacitance. A table like this would be very convenient for a digital circuit designer. The drawback of lookup tables is the work needed to create these tables. Another drawback is that it is not very intuitive. The physical properties of the circuit are hidden underneath the data.

Logical Effort

Another method of sizing CMOS gates is to use the Logical Effort model described in [2]. This model takes the form of a linear equation, which makes it very simple and intuitive for a digital circuit designer to use. It not only can be used to design individual gates, but it can also be used to design a path of gates. But because of its

simplicity, its accuracy pays the price. In order for this model to be accurate, it needs to be calibrated using SPICE simulation data.

Logical Effort for a gate is a number that describes how poor the gate is at driving a load compared to an inverter driving the same load. The inverter is the most basic gate in CMOS and has the most drive capability. That is why it is the reference for Logical Effort. When modeling using Logical Effort, the delay is normalized to the basic delay unit, τ, which is process dependent. It is the delay of an inverter driving an identical inverter without any parasitic capacitance. The normalized unit less delay is d, while the absolute delay is d_{abs}.

$$d_{abs} = d \cdot \tau \tag{5}$$

$$d = f + p \tag{6}$$

f in (6) is the effort delay of a gate, and p is the parasitic delay of a gate. The effort delay consists of two parts: the Logical Effort, g, and the electrical effort, h.

$$f = g \cdot h \tag{7}$$

The electrical effort characterizes the load. It describes how the size of the gate determines its driving capability. Equation (8) defines h with C_{out} being the capacitance loading the gate and C_{in} being the capacitance at the input of the gate.

$$h = \frac{Cout}{Cin} \tag{8}$$

The Logical Effort characterizes the topology of the gate. It describes how the topology of the gate determines its driving capability. Combining (6) and (7) yields the delay of a gate in units of τ.

$$d = g \cdot h + p \tag{9}$$

An inverter has a Logical Effort of one by definition. A list of the Logical Efforts of other gates is shown in Table 1.

Table 1. Logical Effort for static CMOS gates.

	Number of Inputs					
Gate type	**1**	**2**	**3**	**4**	**5**	***n***
Inverter	1					
NAND		4/3	5/3	6/3	7/3	$(n+2)/3$
NOR		5/3	7/3	9/4	11/3	$(2 \cdot n+1)/3$
Multiplexer		2	2	2	2	2
XOR (parity)		4	12	32		

The numbers in Table 1 were derived on the assumption that PMOS transistors have half of the current drive of NMOS transistors, and that the input is a perfect step function. The numbers also neglect velocity saturation and body effects.

The parasitic delay is the delay of a gate due to its internal parasitic capacitances. The major contribution is the drain diffusion capacitances. Table 2 shows a list of parasitic delays for various gates, p_{inv} being the parasitic delay of an inverter. Table 2 exhibits the same assumptions as Table 1. These assumptions will cause discrepancies between the calculated delay and simulated delay. More accurate numbers for Logical Effort and parasitic delay can be obtained by calibrating the numbers through SPICE simulations.

Table 2. Parasitic delays for static CMOS gates.

Gate type	Parasitic Delay
Inverter	p_{inv}
n-input NAND	$n \cdot p_{inv}$
n-input NOR	$n \cdot p_{inv}$
n-way multiplexer	$2 \cdot n \cdot p_{inv}$
XOR, XNOR	$4 \cdot p_{inv}$

To calibrate the numbers in Tables 1 and 2, the propagation delay of the gates is simulated in SPICE with various load capacitances. For an inverter, the delay in picoseconds is plotted with respect to load, h, as in Figure 8. This inverter has an NMOS gate width of 6μm and a PMOS gate width of 10.39μm. The simulations are done with the AMI 0.6μm CMOS process.

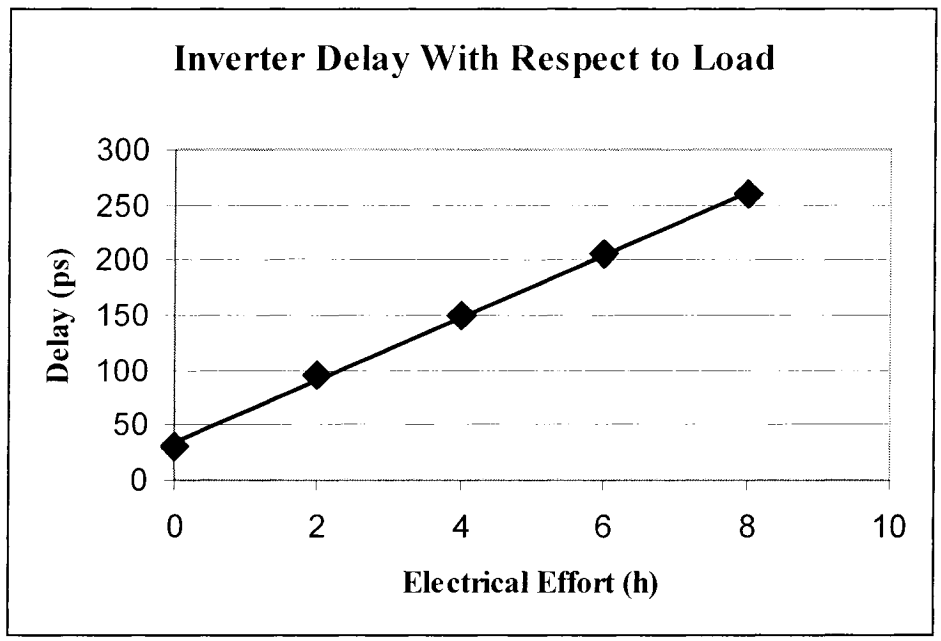

Figure 8. Simulated delay of an inverter driving various loads.

A straight line is fitted to the data. The slope of this line is τ. It intercepts the y-axis at $d = \tau \cdot p_{inv}$. From here, τ and p_{inv} can be calculated. For other gates, the delay is plotted in

a similar fashion but in units of τ as in Figure 9. This NAND2 has an NMOS gate width of 6μm and a PMOS gate width of 8.02μm. A straight line is fitted to the data. The slope of the line, in this case, represents the gate's Logical Effort while the intercept represents the gate's parasitic delay.

Figures 8 and 9 show five data points each for calibrating the Logical Effort model, but more data points would make the calibrated model more accurate to simulation data. A couple of things that make this model inaccurate are the lack of input

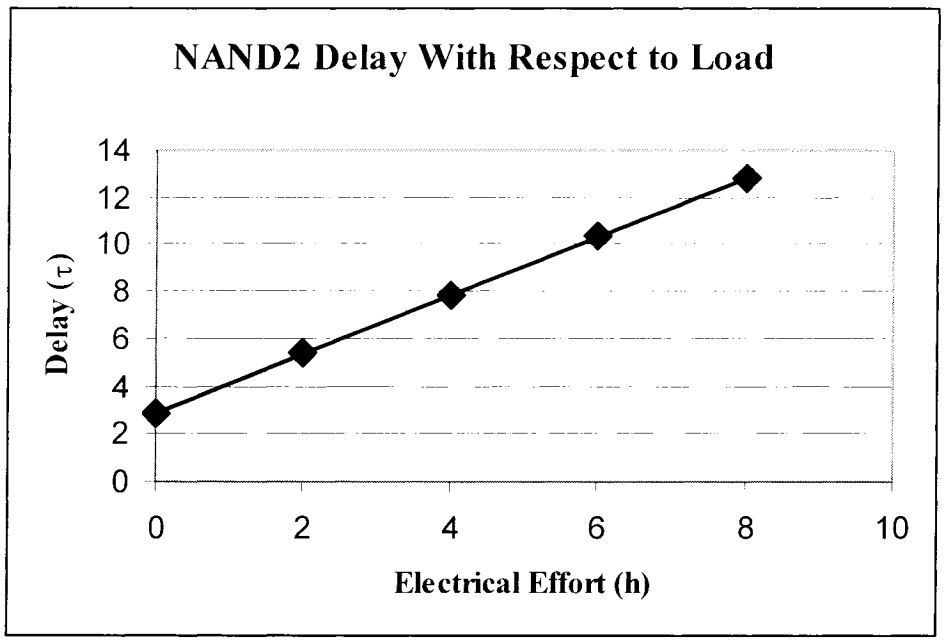

Figure 9. Simulated delay of a NAND2 driving various loads.

slew rate dependence and the constant transistor sizes. These calibrations are done with a constant input slew rate. If the input slew rate were to change, the model will exhibit errors. These calibrations are also done with set transistor sizes. If the transistor sizes of these gates are changed, then the numbers for Logical Effort and parasitic delay will also change. If this model is to be more accurate, calibrations would also need to be done for

various input slew rates and various transistor sizes. This would increase the total number of simulations needed for calibration.

Kang and Leblebici Model

The propagation delay model used in [3] is the model that the work of this thesis is based on. It is a simple and analytical model that can be used to design digital gates. The simplicity of the model takes away from its accuracy. In order for this model to be more accurate, it needs to be calibrated using SPICE simulations.

Kang and Leblebici model the propagation delay of an inverter as

$$\tau_{PHL} = \frac{A \cdot C_{LOAD} \cdot L_N}{W_N} \qquad A = \frac{1}{K_{NP} \cdot (V_{DD} - V_{TN})} \left[\frac{2 \cdot V_{TN}}{V_{DD} - V_{TN}} + \ln\left(\frac{4 \cdot (V_{DD} - V_{TN})}{V_{DD}} - 1 \right) \right] \quad (10)$$

and

$$\tau_{PLH} = \frac{B \cdot C_{LOAD} \cdot L_P}{W_P} \quad B = \frac{1}{K_{PP} \cdot (V_{DD} - |V_{TP}|)} \left[\frac{2 \cdot |V_{TP}|}{V_{DD} - |V_{TP}|} + \ln\left(\frac{4 \cdot (V_{DD} - |V_{TP}|)}{V_{DD}} - 1 \right) \right] . \quad (11)$$

W_N and W_P are the gate widths of the NMOS transistor and the PMOS transistor, respectively. L_N and L_P are the gate lengths of the NMOS transistor and the PMOS transistor, respectively. K_{NP} is equal to the electron mobility in the channel of the NMOS multiplied by the gate oxide capacitance. Similarly, K_{PP} is the hole mobility in the channel of the PMOS multiplied by the gate oxide capacitance. V_{TN} and V_{TP} are the threshold voltages for the NMOS and PMOS respectively. C_{LOAD} is the total capacitance loading the inverter, including internal and external capacitances, as shown in Figure 10. The capacitances making up C_{LOAD} are given below.

$$C_{LOAD} = C_G + C_{INT} + C_{OUT} \quad (12)$$

C_G is the input capacitance to the gate or gates loading the inverter. C_{INT} is the interconnect capacitance, and C_{OUT} is the internal capacitance at the output node of the inverter. C_{OUT} consists of the drain to body capacitances as well as the gate to drain capacitances.

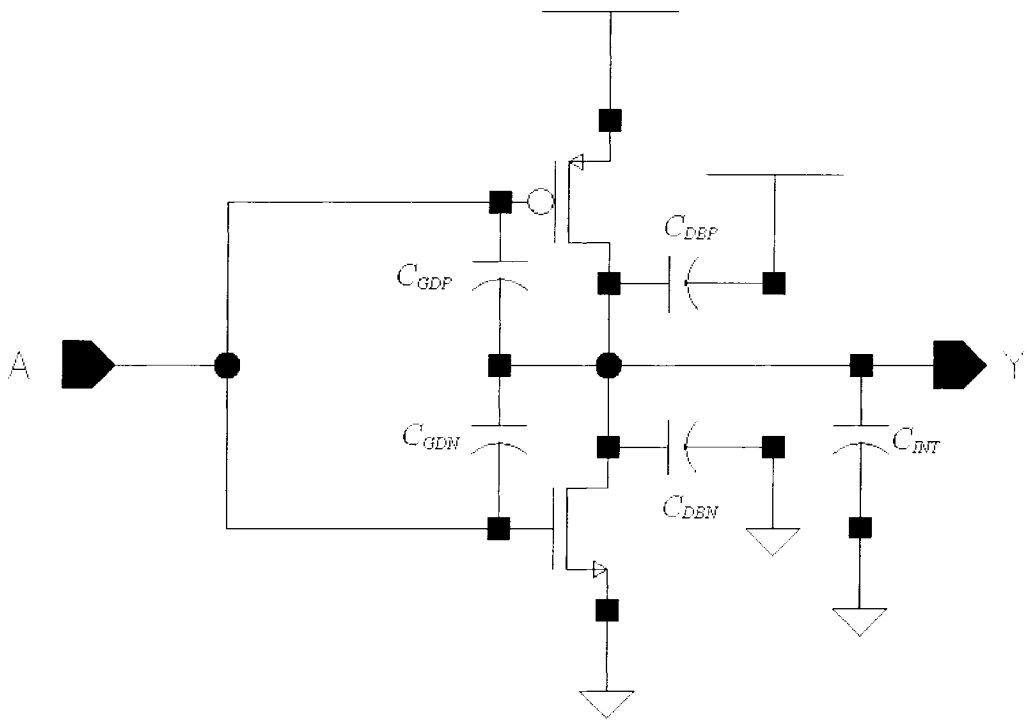

Figure 10. Inverter with capacitances explicitly shown.

$$C_{OUT} = C_{DBN} + C_{DBP} + 2 \cdot C_{GDN} + 2 \cdot C_{GDP} \tag{13}$$

$$C_{GDN} = C_{GDON} \cdot W_N \tag{14}$$

$$C_{GDP} = C_{GDOP} \cdot W_P \tag{15}$$

$$\begin{aligned} C_{DBN} &= C_{JN} \cdot D_{DRAIN} \cdot W_N \cdot K_{EQJN} + \\ &C_{JSWN} \cdot (2 \cdot D_{DRAIN} + W_N) \cdot K_{EQSWN} + C_{JSWGN} \cdot W_N \cdot K_{EQSWGN} \end{aligned} \tag{16}$$

$$\begin{aligned} C_{DBP} &= C_{JP} \cdot D_{DRAIN} \cdot W_P \cdot K_{EQJP} + \\ &C_{JSWP} \cdot (2 \cdot D_{DRAIN} + W_P) \cdot K_{EQSWP} + C_{JSWGP} \cdot W_P \cdot K_{EQSWGP} \end{aligned} \tag{17}$$

C_{GDN} and C_{GDP} are the gate to drain capacitances of NMOS and PMOS transistors. These capacitances are multiplied by two because that is what the output node effectively sees according to the Miller effect as described in [49]. The gate to drain capacitance is connected to the input on one end and the output on the other end. The input changes from rail to rail, and the output switches from rail to rail in the opposite direction. This leads to an effective voltage change of $2 \cdot V_{DD}$. This doubling of voltage also means a doubling of charge. C_{DBN} and C_{DBP} are the drain to body capacitances of NMOS and PMOS transistors. C_{GDON} and C_{GDOP} are gate to drain overlap capacitances per unit width. C_{JN} and C_{JP} are diffusion capacitances per unit area underneath the drains of NMOS and PMOS transistors. C_{JSWN} and C_{JSWP} are sidewall diffusion capacitances per unit length around the drain areas. C_{JSWGN} and C_{JSWGP} are sidewall diffusion capacitances adjacent to the gate per unit length. These are sometimes different from the other three sidewalls of the drain because they are sometimes lightly doped. D_{DRAIN} is the length of the drain region. K_{EQJN} and K_{EQJP} are voltage equivalence factors for the bottom junctions of NMOS and PMOS drains. K_{EQSWN} and K_{EQSWP} are voltage equivalent factors for the sidewall junctions of NMOS and PMOS drains. K_{EQSWGN} and K_{EQSWGP} are voltage equivalent factors for the sidewall junctions adjacent to the gate of NMOS and PMOS drains. These voltage equivalent factors are used to take into account the variation of junction capacitance with respect to voltage swing. They can be calculated using the following equation.

$$K_{EQ} = \frac{\phi_0}{\frac{V_{DD}}{2} \cdot (1-m)} \cdot \left[\left(1 + \frac{V_{DD}}{2 \cdot \phi_0} \right)^{(1-m)} - 1^{(1-m)} \right] \tag{18}$$

ϕ_0 is the built in potential of the junction, and m is the grading coefficient of the junction.

C_{GDON}, C_{GDOP}, C_{JN}, C_{JP}, C_{JSWN}, C_{JSWP}, C_{JSWGN}, C_{JSWGP}, m_{JN}, m_{JP}, m_{JSWN}, m_{JSWP}, m_{JSWGN}, m_{JSWGP},ϕ_{0JN}, ϕ_{0JP}, ϕ_{0JSWN}, ϕ_{0JSWP}, ϕ_{0JSWGN}, and ϕ_{0JSWGP} can be found in SPICE models. They are in bold letters in the example SPICE models shown in Figures 11 and 12. Sometimes the sidewall capacitances will be the same all around the drain. In these cases, C_{JSWGN}, C_{JSWGP}, m_{JSWGN}, m_{JSWGP}, ϕ_{0JSWGN}, and ϕ_{0JSWGP} will not be specified in the model.

For symmetric propagation delays, meaning τ_{PHL} is equal to τ_{PLH}, the delay equations are set equal to each other.

$$\frac{B \cdot C_{LOAD} \cdot L_P}{W_P} = \frac{A \cdot C_{LOAD} \cdot L_N}{W_N} \tag{19}$$

Knowing that L_P is equal to L_N and C_{LOAD} is constant, it can be seen that

$$\frac{B}{A} = \frac{WP}{WN}. \tag{20}$$

This ratio of W_P/W_N will give symmetric propagation delays. It will be known as R from here on.

When sizing digital gates, it is convenient to see W_N as a function of propagation delay and load capacitance. This can be done by combining (10) with R.

$$W_N = \frac{C_G + C_{INT} + 2 \cdot (C_{JSWN} \cdot D_{DRAIN} \cdot K_{EQSWN} + C_{JSWP} \cdot D_{DRAIN} \cdot K_{EQSWP})}{\frac{\tau_{PHL}}{A \cdot L_N} - \beta} \tag{21}$$

β here is the capacitance that is multiplied by transistor sizes.

$$\begin{aligned}\beta = {} & C_{GDON} + C_{GDOP} \cdot R + C_{JN} \cdot D_{DRAIN} \cdot K_{EQJN} \\ & + C_{JSWN} \cdot K_{EQSWN} + C_{JSWNGN} \cdot K_{EQSWGN} + C_{JP} \cdot D_{DRAIN} \cdot R \cdot K_{EQJP} \\ & + C_{JSWP} \cdot R \cdot K_{EQSWP} + C_{JSWGP} \cdot R \cdot K_{EQSWGP}\end{aligned} \tag{22}$$

With W_N known from (21) and (22), W_P can be calculated as follows.

$$W_P = R \cdot W_N . \tag{23}$$

.MODEL tsmc18dN NMOS(LEVEL=11

VERSION=3.1	TNOM=27	TOX=4.1E-9
XJ=1E-7	NCH=2.3549E17	VTH0=0.3680311
K1=0.5912273	K2=2.289277E-3	K3=1E-3
K3B=2.5435988	W0=1E-7	NLX=1.738916E-7
DVT0W=0	DVT1W=0	DVT2W=0
DVT0=1.9209561	DVT1=0.4429618	DVT2=-0.0147589
U0=272.9837819	UA=-1.132212E-9	UB=1.883546E-18
UC=2.04883E-11	VSAT=9.701073E4	A0=1.8787969
AGS=0.3794	B0=-1.850639E-8	B1=-1E-7
KETA=-4.764457E-3	A1=0	A2=1
RDSW=130.862057	PRWG=0.5	PRWB=-0.2
WR=1	WINT=4.679815E-9	LINT=6.412758E-9
XL=-2E-8	XW=-1E-8	DWG=5.911695E-9
DWB=1.005248E-8	VOFF=-0.0861327	NFACTOR=2.1794075
CIT=0	CDSC=2.4E-4	CDSCD=0
CDSCB=0	ETA0=5.643736E-4	ETAB=-1.416818E-3
DSUB=0.0510645	PCLM=1.7575212	PDIBLC1=0.6724008
PDIBLC2=5.221653E-3	PDIBLCB=-0.1	DROUT=0.8603216
PSCBE1=4.803723E10	PSCBE2=4.871276E-8	PVAG=1.0498691
DELTA=0.01	RSH=6.7	MOBMOD=1
PRT=0	UTE=-1.5	KT1=-0.11
KT1L=0	KT2=0.022	UA1=4.31E-9
UB1=-7.61E-18	UC1=-5.6E-11	AT=3.3E4
WL=0	WLN=1	WW=0
WWN=1	WWL=0	LL=0
LLN=1	LW=0	LWN=1
LWL=0	CAPMOD=2	XPART=0.5
CGDO=6.9E-10	CGSO=6.9E-10	CGBO=1E-12
CJ=1.012015E-3	**PB=0.7319956**	**MJ=0.3612578**
CJSW=2.171766E-10	**PBSW=0.6328732**	**MJSW=0.1191771**
CJSWG=3.3E-10	**PBSWG=0.6328732**	**MJSWG=0.1191771**
CF=0	PVTH0=-3.166914E-3	PRDSW=-1.7313214
PK2=1.550245E-3	WKETA=-1.876456E-3	LKETA=2.206463E-3
PU0=-2.6252751	PUA=-3.86698E-11	PUB=0
PVSAT=1.081788E3	PETA0=1E-4	PKETA=1.035013E-3)

Figure 11. Example SPICE model of an NMOS transistor.

.MODEL tsmc18dP PMOS (LEVEL=11

VERSION=3.1	TNOM=27	TOX=4.1E-9
XJ=1E-7	NCH=4.1589E17	VTH0=-0.4149264
K1=0.599378	K2=0.0233531	K3=0
K3B=13.5464133	W0=1E-6	NLX=7.089213E-8
DVT0W=0	DVT1W=0	DVT2W=0
DVT0=0.4369111	DVT1=0.2835489	DVT2=0.0831221
U0=120.9245479	UA=1.722692E-9	UB=1.460812E-21
UC=-9.3774E-11	VSAT=2E5	A0=1.5766852
AGS=0.3767784	B1=5E-6	B0=1.019319E-6
B1=5E-6	KETA=0.0186763	A1=0.4410437
RDSW=244.3589747	A2=0.3002749	RDSW=244.3589747
PRWG=0.5	PRWB=-0.5	WR=1
WINT=3.424328E-9	LINT=2.038046E-8	XL=-2E-8
XW=-1E-8	DWG=-1.520363E-8	DWB=-1.879906E-8
VOFF=-0.1033304	NFACTOR=1.796676	CIT=0
CDSC=2.4E-4	CDSCB=0	CDSCD=0
CDSCB=0	ETA0=0.0283932	ETAB=-0.0938944
DSUB=0.7599844	PCLM=1.5542195	PDIBLC1=1.304291E-3
PDIBLC2=0.0163173	PDIBLCB=-1E-3	DROUT=0
PSCBE1=1.842459E9	PSCBE2=5.321454E-10	PVAG=6.3532208
DELTA=0.01	RSH=7.5	MOBMOD=1
PRT=0	UTE=-1.5	KT1=-0.11
KT1L=0	KT2=0.022	UA1=4.31E-9
UB1=-7.61E-18	UC1=-5.6E-11	AT=3.3E4
WL=0	WLN=1	WW=0
WWN=1	WWL=0	LL=0
LLN=1	LW=0	LWN=1
LWL=0	CAPMOD=2	XPART=0.5
CGDO=6.86E-10	CGSO=6.86E-10	CGBO=1E-12
CJ=1.135715E-3	**PB=0.8647945**	**MJ=0.4227781**
CJSW=1.878779E-10	**PBSW=0.6209079**	**MJSW=0.2821903**
CJSWG=4.22E-10	**PBSWG=0.6209079**	**MJSWG=0.2821903**
PRDSW=7.5737967	CF=0	PVTH0=8.450209E-4
PRDSW=7.5737967	PK2=1.878009E-3	WKETA=2.478813E-3
LKETA=-1.186131E-3	PU0=-1.7399784	PUA=-7.09196E-11
PUB=1E-21	PVSAT=50	PETA0=1E-4
PKETA=3.72283E-3)		

Figure 12. Example SPICE model of a PMOS transistor.

For NAND and NOR gates, there are more capacitances contributing to the propagation delay. These capacitances can be more complex to model because they are not all in parallel. Let us take the NAND4 for example. It is shown in Figure 13 with its internal capacitances explicitly shown. The PMOS gate to drain and drain to body capacitances are connected in parallel, so they can simply be added together. The NMOS gate to drain and drain to body capacitances, on the other hand, are not all connected in parallel so they cannot just be added together. These capacitances would have to be calculated in a different way.

If we model the path from the output to ground as an *RC* ladder network, as in Figure 14, we can use a special case of the Elmore delay as described in [3]. This special case of the Elmore delay is when there is only one path of resistors. If a step voltage is applied to the input, the delay that the output will have can be modeled as

$$\begin{aligned} \tau_D = {} & R_1 \cdot C_1 + (R_1 + R_2) \cdot C_2 + (R_1 + R_2 + R_3) \cdot C_3 + \\ & \cdots + (R_1 + R_2 + R_3 + \cdots + R_N) \cdot C_N \end{aligned} \tag{24}$$

or

$$\tau_D = \sum_{j=1}^{N} C_j \sum_{k=1}^{j} R_k \,. \tag{25}$$

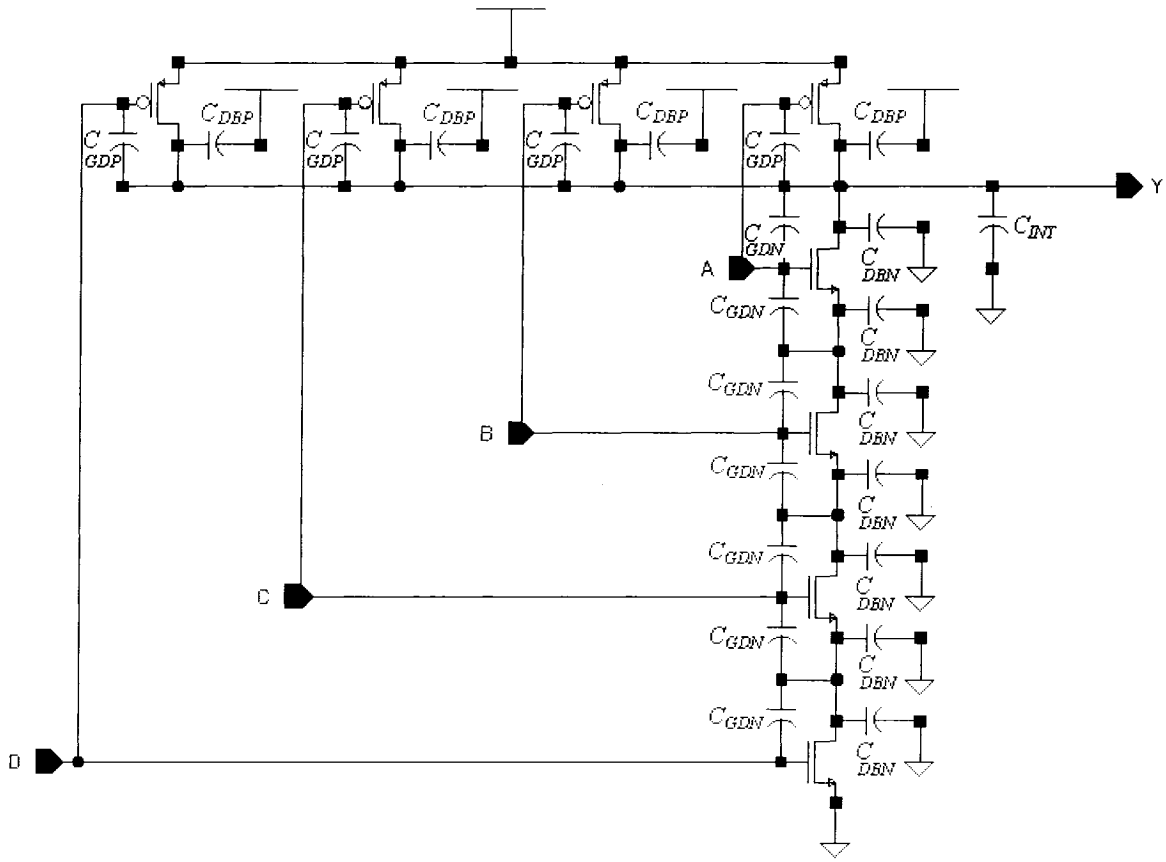

Figure 13. NAND4 with internal capacitances explicitly shown.

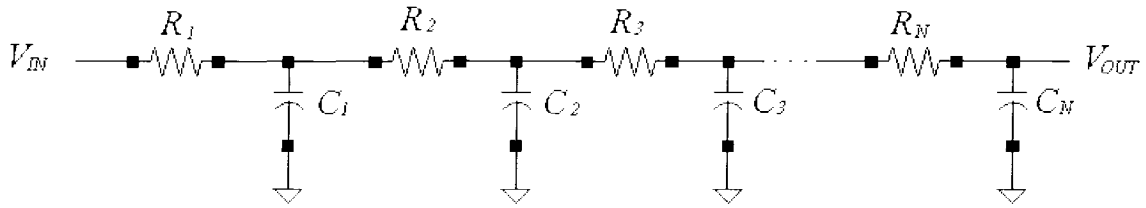

Figure 14. RC ladder network.

Applying this delay model to the NAND4 and assuming that the total resistance in the path is R, with each resistor being equal to $R/4$, we get

$$\tau_{PHL} = C_1 \cdot \frac{R}{4} + C_2 \cdot 2 \cdot \frac{R}{4} + C_3 \cdot 3 \cdot \frac{R}{4} + C_4 \cdot 4 \cdot \frac{R}{4}. \tag{26}$$

$$C_1 = 3 \cdot C_{GDN} + 2 \cdot C_{DBN} \tag{27}$$

$$C_2 = 2 \cdot C_{GDN} + 2 \cdot C_{DBN} \tag{28}$$

$$C_3 = 2 \cdot C_{GDN} + 2 \cdot C_{DBN} \tag{29}$$

$$C_4 = C_{GDN} + 5 \cdot C_{GDP} + C_{DBN} + 4 \cdot C_{DBP} + C_G + C_{INT} \tag{30}$$

C_1 is the capacitance on the node just above the bottom transistor of the NAND4, and C_4 is the capacitance on the output node. C_2 and C_3 are the capacitances at the nodes in between. The extra C_{GDN} in C_1 is due to the Miller effect. Here we are using the bottom transistor as the switching transistor. Similarly, the extra C_{GDP} in C_4 is due to the Miller effect as well. Combining (26)-(30) and simplifying, we get

$$\tau_{PHL} = (4 \cdot C_{DBN} + 4 \cdot C_{DBP} + 4.25 \cdot C_{GDN} + 5 \cdot C_{GDP} + C_G + C_{INT}) \cdot R\,. \tag{31}$$

This analysis can be extended for NAND gates in general with the following equation.

$$\tau_{PHL} = \left[\begin{array}{l} N_{SN} \cdot (C_{DBN} + C_{DBP}) + \left(N_{SN} + \dfrac{1}{N_{SN}} \right) \cdot C_{GDN} \\ + (N_{SN} + 1) \cdot C_{GDP} + C_G + C_{INT} \end{array} \right] \cdot R \tag{32}$$

Where N_{SN} is the number of series connected NMOS transistors in the NAND gate. For instance, for a NAND3, N_{SN} is 3.

If this same analysis were done for NOR gates, the delay would be

$$\tau_{PLH} = \left[\begin{array}{l} N_{SP} \cdot (C_{DBN} + C_{DBP}) + (N_{SP} + 1) \cdot C_{GDN} \\ + \left(N_{SP} + \dfrac{1}{N_{SP}} \right) \cdot C_{GDP} + C_G + C_{INT} \end{array} \right] \cdot R\,. \tag{33}$$

Where N_{SP} is the number of series connected PMOS transistors in the NOR gate.

Looking at (32) and (33), it can be seen that the load capacitance can be modeled as

$$C_{LOAD} = N \cdot (C_{DBN} + C_{DBP}) + \left(N + \frac{1}{N_{SN}} \right) \cdot C_{GDN} + \left(M + \frac{1}{N_{SP}} \right) \cdot C_{GDP} + C_G + C_{INT} \quad (34)$$

For NAND gates, both N and M equal N_{SN}. For NOR gates, N and M equal N_{SP}. NAND and NOR gates are special cases. In general, N and M are the weighted number of NMOS drains and PMOS drains, respectively, in a gate. Each node in a stack of transistors has its own weight. That weight depends on how high the stack is and how far from the output the node is. For instance, the NAND4 has four nodes in the stack of NMOS transistors that are not connected to ground. Let the node furthest from the output be node 1 and the output node be node 4. The drains at node 1 are weighted as $1/N_{SN}$, or 1/4 in this case. There are two drains at node 1, so there are 2/4 weighted drains. Similarly, there are $2 \cdot 2/4=1$ weighted drains at node 2, $2 \cdot 3/4=3/2$ at node 3, and $1 \cdot 4/4=1$ at node 4. In total, there are $1/2+1+3/2+1=4$ weighted drains in the NMOS stack of a NAND4. This is the N of the NAND4. Also in a NAND4, there are four PMOS stacks that are each one transistor high. Each drain, in this case, are weighted as 1/1. There are four drains each weighted at 1/1, which equals 4 for M. In summary, to calculate the number of weighted drains on a node, multiply the total number of drains on the node by the weight, which is the node number divided by the height of the stack of transistors. N is the total number of weighted NMOS drains, and M is the total number of PMOS drains.

Expanding (34) to include the SPICE capacitance parameters yields the following.

$$C_{LOAD} = W_N \cdot \left[\begin{array}{l} C_{GDON} \cdot \left(N + \frac{1}{N_{SN}} \right) + C_{GDOP} \cdot R \cdot \left(M + \frac{1}{N_{SP}} \right) \\ + N \cdot (C_{JN} \cdot D_{DRAIN} \cdot K_{EQJN} + C_{JSWN} \cdot K_{EQSWN} \\ + C_{JSWGN} \cdot K_{EQSWGN}) + M \cdot (C_{JP} \cdot D_{DRAIN} \cdot R \cdot K_{EQJP} \\ + C_{JSWP} \cdot R \cdot K_{EQSWP} + C_{JSWGP} \cdot R \cdot K_{EQSWGP}) \end{array} \right] + \tag{35}$$
$$(N + M) \cdot (C_{JSWN} \cdot D_{DRAIN} \cdot K_{EQSWN} + C_{JSWP} \cdot D_{DRAIN} \cdot K_{EQSWP}) + C_G + C_{INT}$$

Like (22), the transistor size dependent capacitance can be specified.

$$\begin{array}{l} \beta = C_{GDON} \cdot \left(N + \frac{1}{N_{SN}} \right) + C_{GDOP} \cdot R \cdot \left(M + \frac{1}{N_{SP}} \right) \\ + N \cdot (C_{JN} \cdot D_{DRAIN} \cdot K_{EQJN} + C_{JSWN} \cdot K_{EQSWN} \\ + C_{JSWGN} \cdot K_{EQSWGN}) + M \cdot (C_{JP} \cdot D_{DRAIN} \cdot R \cdot K_{EQJP} \\ + C_{JSWP} \cdot R \cdot K_{EQSWP} + C_{JSWGP} \cdot R \cdot K_{EQSWGP}) \end{array} \tag{36}$$

Using (35) and (36), (21) can be adjusted to include NAND and NOR gates.

$$W_N = \frac{C_G + C_{INT} + (N + M) \cdot \left(\begin{array}{l} C_{JSWN} \cdot D_{DRAIN} \cdot K_{EQSWN} \\ + C_{JSWP} \cdot D_{DRAIN} \cdot K_{EQSWP} \end{array} \right)}{\frac{\tau_{PHL}}{A \cdot L_N \cdot N_{SN}} - \beta} \tag{37}$$

L_N is multiplied by N_{SN} to account for the effective channel length of the gate. The current drive decreases with an increase in N_{SN}. These series connected transistors also affect the ratio for symmetric propagation delay, R.

$$R = \frac{B \cdot N_{SP}}{A \cdot N_{SN}} \tag{38}$$

This model is very simple and intuitive, but the simplifications made cause large discrepancies with simulation results. This model is derived using the square law current equations along with the assumption that the input makes a transition instantaneously.

That means that this model does not take into account velocity saturation, input slew rate, body effect, or channel length modulation. To increase the accuracy of this model, A and R need to be calibrated from SPICE simulations. The calibration of an inverter is illustrated below. This calibration method can also be done for other gates in the same fashion.

To find a reasonable propagation delay to extract A and R with, a ring oscillator is used. An example ring oscillator is shown in Figure 15.

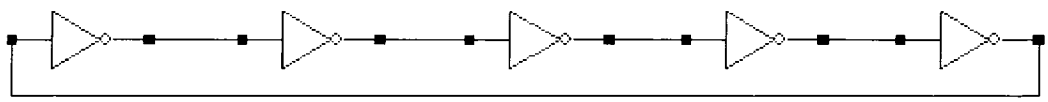

Figure 15. Schematic of a ring oscillator used to find a reasonable propagation delay for calibrating A and R.

W_N is set to minimum width and W_P is set to twice the minimum width. The following equation from [3] is used to calculate the propagation delay.

$$\tau = \frac{1}{2 \cdot f \cdot n} \tag{39}$$

τ is the propagation delay of each inverter, f is the frequency of the ring oscillator, and n is the number of stages in the ring oscillator. This is repeated with W_N set to 25 times the channel length and W_P set to 50 times the channel length. The minimum of these two propagation delays is used for calibration.

According to [50], fanout-of-four (*FO4*) inverters is typically the maximum load a logic gate would drive. Figure 16 shows a schematic of fanout-of-four inverters.

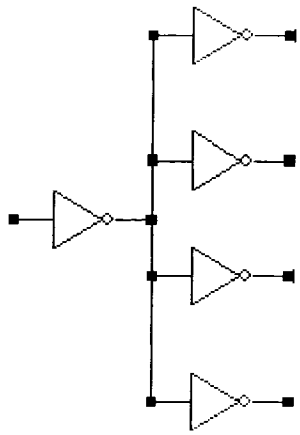

Figure 16. *FO4* inverters load.

A *FO1* inverter load is also used quite often. It is through my experience that a *FO2* inverters load is the optimum load for calibration. The load is calculated using inverters with NMOS transistors with two times the minimum gate width and PMOS transistors with four times the minimum gate width.

A test bench is set up similar to Figure 17. The *FO2* load, the minimum propagation delay from the ring oscillator simulations, A calculated from (10), and an initial guess of two for R are used in (21) to find an initial W_N and an initial W_P. A pulse voltage source, with a rise and fall time of ten picoseconds likely that the propagation delays will neither be equal to the target nor be equal to each other. To adjust W_N and W_P to get closer to the target delay, the following equations are used.

$$W_N = \frac{W'_N \cdot \tau_{PHL\_M}}{\tau_{PHL}} \tag{40}$$

$$W_P = \frac{W'_P \cdot \tau_{PLH\_M}}{\tau_{PLH}} \tag{41}$$

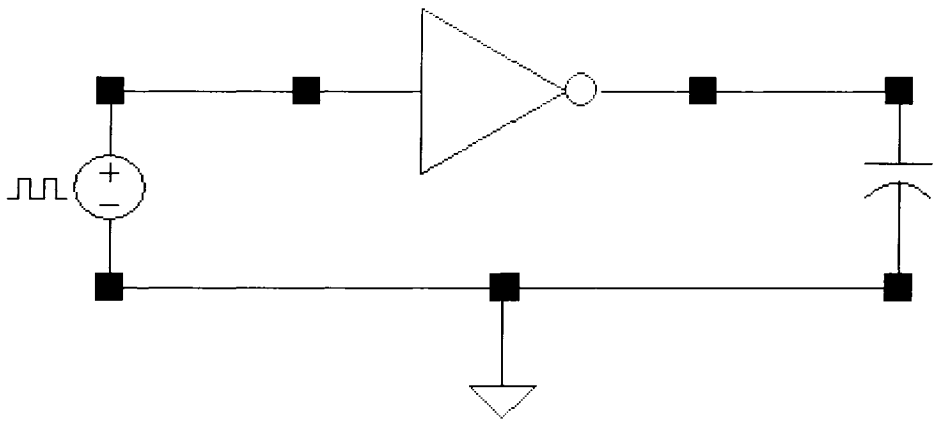

Figure 17. Example test bench to calibrate A and R.

τ_{PHL_M} and τ_{PLH_M} are the measured delays while τ_{PHL} and τ_{PLH} are the target delays. W'_N and W'_P are the previous widths. The adjusted values of W_N and W_P are entered into the test bench and simulated. This process is repeated until the delays reach the target within one percent error. Typically, this process would take about ten simulation runs. R is then calculated from these final values of W_N and W_P. They are inserted back into (21), and A is calculated. This whole process gives the calibrated values of A and R for an inverter. These numbers will not be accurate when used with other gates. A and R need to be calibrated in the same fashion for other gates as well.

An example of this calibration process is shown in Table 3. In this example, an inverter is calibrated in the TSMC 0.18μm technology. A propagation delay of 32.3ps is calculated from the ring oscillator simulation. The *FO2* load is calculated to be 7.14fF. The initial A is calculated from (10) to be 9830Ω. Along with the initial guess of 2 for R, W_N and W_P is calculated to be 0.484μm and 0.968μm respectively. These initial conditions are entered into a test bench as in Figure 14, and the propagation delays are measured. The error is calculated and the new transistor widths are calculated using (27)

and (28). As can be seen in Table 3, seven simulations are needed to get the measured propagation delay within one percent of the target delay. Taking the final W_N and W_P, R is calculated to be 2.2205. Inserting these numbers back into (21) and calculating for A yields 14059Ω.

These calibrations still do not account for the dependence of propagation delay on input slew rate because they are calibrated using a pulse voltage source with only ten picoseconds of input rise and fall times. In order for this model to include input slew rate dependence, it needs to be calibrated for various input slew rates. These calibrations can be very time consuming.

Table 3. Example numbers from calibration simulations done for an inverter in TSMC 0.18μm technology.

Simulation			$\tau_{PHL}(ps)$	**32.3**	**% Error**		
1	$W'_N(cm)$	4.84E-05	$\tau_{PHL\_M}(ps)$	37.8	17.03	$W_N(cm)$	5.66E-05
	$W'_P(cm)$	9.68E-05	$\tau_{PLH\_M}(ps)$	44.2	36.84	$W_P(cm)$	1.32E-04
2	$W'_N(cm)$	5.66E-05	$\tau_{PHL\_M}(ps)$	37	14.55	$W_N(cm)$	6.49E-05
	$W'_P(cm)$	1.32E-04	$\tau_{PLH\_M}(ps)$	36.5	13.00	$W_P(cm)$	1.50E-04
3	$W'_N(cm)$	6.49E-05	$\tau_{PHL\_M}(ps)$	35	8.36	$W_N(cm)$	7.03E-05
	$W'_P(cm)$	1.50E-04	$\tau_{PLH\_M}(ps)$	34.4	6.50	$W_P(cm)$	1.59E-04
4	$W'_N(cm)$	7.03E-05	$\tau_{PHL\_M}(ps)$	33.9	4.95	$W_N(cm)$	7.38E-05
	$W'_P(cm)$	1.59E-04	$\tau_{PLH\_M}(ps)$	33.4	3.41	$W_P(cm)$	1.65E-04
5	$W'_N(cm)$	7.38E-05	$\tau_{PHL\_M}(ps)$	33.1	2.48	$W_N(cm)$	7.56E-05
	$W'_P(cm)$	1.65E-04	$\tau_{PLH\_M}(ps)$	33	2.17	$W_P(cm)$	1.68E-04
6	$W'_N(cm)$	7.56E-05	$\tau_{PHL\_M}(ps)$	32.8	1.55	$W_N(cm)$	7.68E-05
	$W'_P(cm)$	1.68E-04	$\tau_{PLH\_M}(ps)$	32.7	1.24	$W_P(cm)$	1.71E-04
7	**$W'_N(cm)$**	**7.68E-05**	**$\tau_{PHL\_M}(ps)$**	**32.6**	**0.93**		
	$W'_P(cm)$	**1.71E-04**	**$\tau_{PLH\_M}(ps)$**	**32.6**	**0.93**		

Method

Calibrations can take excessive time. Each gate takes approximately seven to ten simulate and tweak runs to calibrate. If two, three, and four input NAND and NOR gates are to be calibrated, that would be between forty-two to sixty simulations. In this chapter, with just the calibrated inverter and four dc sweep simulations, A and R can be calibrated for NAND and NOR gates.

Calibration For Fast Inputs

When the inverter was calibrated in the previous chapter, an input pulse voltage with 10ps rise and fall time was used. In reality, 10ps is too fast for any practical circuit. Practical rise and fall times are different for each technology. To find the fastest practical input rise and fall time, a chain of inverters is analyzed as in Figure 18.

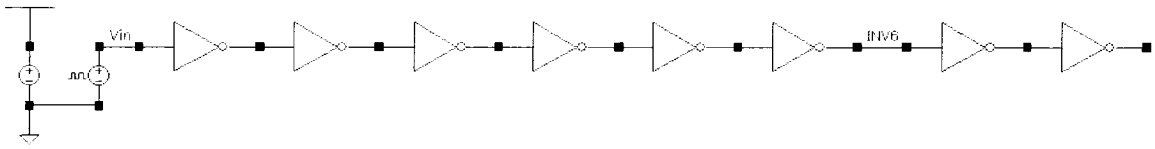

Figure 18. Chain of inverters used to find fastest rise and fall time.

Each inverter is sized to have an NMOS that is twice the minimum width and a PMOS that is four times the minimum width. A pulse voltage with 10ps rise and fall time is applied to the input inverter. As the signal propagates through the chain, the rise and fall times will get slower, but eventually they will stop slowing down. For instance, the fifth inverter will have the same rise and fall time transition on its output as the sixth inverter down the chain. These rise and fall times are the fastest practical rise and fall times within a technology. It is the minimum of the rise and fall times at the output of the sixth

inverter that the inverter is calibrated for. The seventh and eight inverters in the chain are used to ensure a realistic load.

The procedure for calibrating the inverter is the same as the one described in the previous chapter for the Kang and Leblebici model except that the pulse voltage has the practical input rise and fall times.

When calibrating A and R, what are really being calibrated are the resistances of the gates. A is the normalized resistance of the NMOS transistors, and B is the normalized resistance of the PMOS transistors. R is the ratio of the two.

$$R = \frac{B}{A} \tag{42}$$

Equation (37) assumes that the A of the inverter can be used for NAND gates if L_N is multiplied by N_{SN}, $L_N \cdot N_{SN}$. This estimation is inaccurate because of carrier velocity saturation. For an inverter, all of the voltage from output to ground is applied across just one NMOS transistor. This causes extreme velocity saturation. For a NAND gate, the voltage from the output to ground gets shared among multiple transistors. This makes NAND gates not as velocity saturated as inverters. For this same reason, channel length modulation plays less of a factor for NAND gates than for inverters. The total resistance of NAND gates is not simply $A \cdot N_{SN}$, it will be less than that. If (37) is to be used, A would need to be multiplied by some factor less than one.

By looking at a dc simulation of stacked NMOS transistors, we will be able to find what that factor is for NAND gates. A schematic such as the one in Figure 19 is

used for this analysis. All transistors are sized the same width as the NMOS of the calibrated inverter.

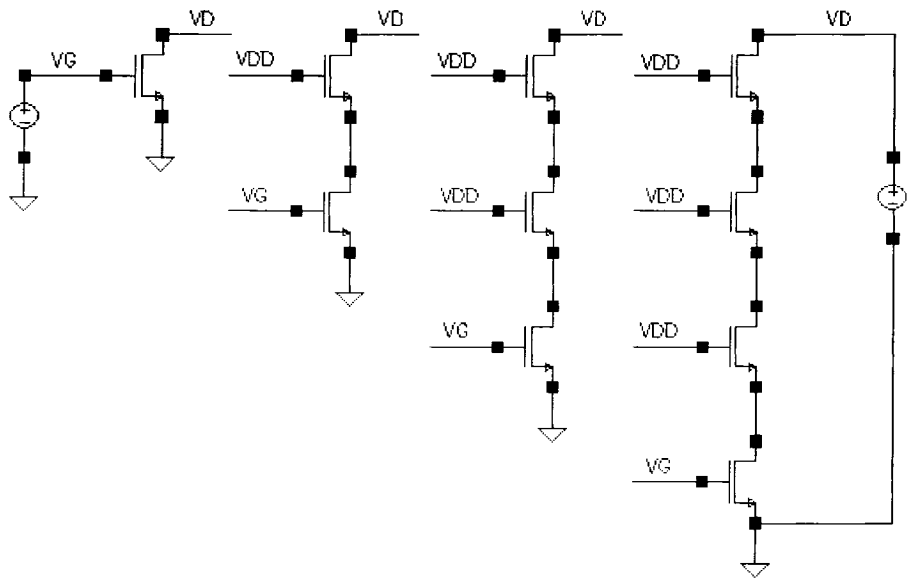

Figure 19. Stacked NMOS dc sweep simulation schematic.

This schematic is set up to emulate the stack of NMOS transistors in a NAND gate with the bottom transistor switching. For a fast but finite input slew rate, the input does not reach V_{DD} instantaneously on the rising edge. It takes some time to get there, and it spends some time at lower voltages. To emulate this, V_G is set to 90% of V_{DD}. The voltage range on the output that we are concerned with for the propagation delay is from V_{DD} down to $V_{DD}/2$. To emulate this, V_D is swept from $V_{DD}/2$ to V_{DD}. The resistance of each stack can then be calculated as follows.

$$RES(N_{SN}) = \int_{\frac{V_{DD}}{2}}^{V_{DD}} \frac{1}{I(N_{SN})} dV_D \tag{43}$$

Where $RES(N_{SN})$ is the resistance of the N_{SN} high stack, and $I(N_{SN})$ is the current through the N_{SN} high stack. From these resistances, the factor that A needs to be multiplied by can be found.

$$\gamma = \frac{RES(N_{SN})}{RES(N_{SN}=1)\cdot N_{SN}} \tag{44}$$

The high to low propagation delay can now be adjusted as follows.

$$\tau_{PHL} = \frac{A\cdot\gamma\cdot C_{LOAD}\cdot L_N\cdot N_{SN}}{W_N} \tag{45}$$

B for NAND gates can be estimated to be the same as the B for inverters because the PMOS stack stays equal to one PMOS transistor. So we can simply use the resistances of the NMOS stacks to estimate R.

$$\frac{R_{NAND}}{R_{INV}} = \frac{\frac{B}{A\cdot\gamma\cdot N_{SN}}}{\frac{B}{A}} \tag{46}$$

Where R_{NAND} is the R of the NAND gate, and R_{INV} is the R of the calibrated inverter. It can be seen that the B's and A's in (46) cancel.

$$R_{NAND} = \frac{R_{INV}}{\gamma\cdot N_{SN}} \tag{47}$$

For NOR gates, A stays constant because the NMOS stack stays at one transistor. The B, on the other hand, will change because the PMOS stack increases. The factor to multiply B by can be found in a similar fashion as that for A of NAND gates. In this case, stacked PMOS transistors are set up in a dc simulation as in Figure 20.

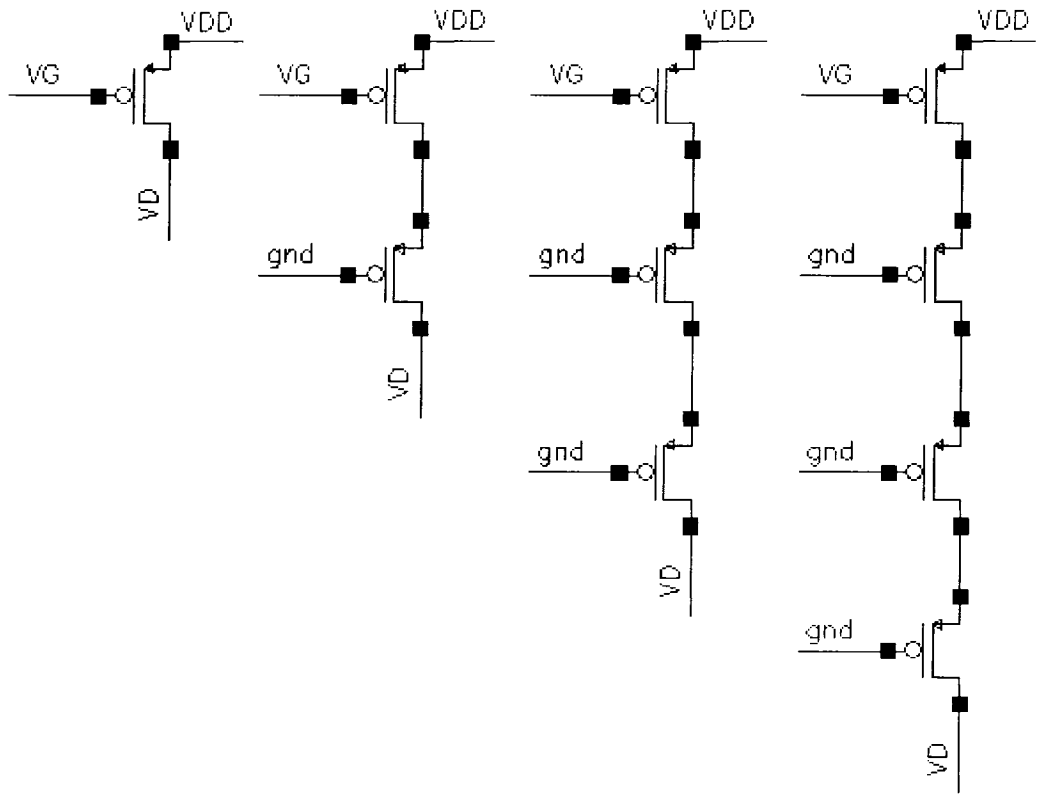

Figure 20. Stacked PMOS dc sweep simulation schematic.

The transistors are the same size as the PMOS transistor of the calibrated inverter. This schematic is set up to emulate the PMOS stack of NOR gates with the top transistor switching. A fast but finite input slew rate does not reach ground instantaneously. It takes some time to get there, and it spends some time at higher voltages. That is why V_G is set to 10% of V_{DD}. For the low to high propagation delay of NOR gates, the output goes from ground to $V_{DD}/2$. Similarly, in this dc simulation, V_D is swept from zero volts to $V_{DD}/2$. The resistances can be found by integrating the inverse of the current over the swept voltage range.

$$RES(NSP) = \int_0^{VDD/2} \frac{1}{I(NSP)} dVD \tag{48}$$

From the resistances, the factor to multiply B by can be found.

$$\gamma = \frac{RES(N_{SP})}{RES(N_{SP}=1)\cdot N_{SP}} \tag{49}$$

The low to high propagation delay can be adjusted to include γ.

$$\tau_{PLH} = \frac{B\cdot\gamma\cdot C_{LOAD}\cdot L_P\cdot N_{SP}}{W_P} \tag{50}$$

The high to low propagation delay stays the same as (10) because N_{SN} remains at one, but γ can be used to calculate R for NOR gates.

$$\frac{R_{NOR}}{R_{INV}} = \frac{\frac{B\cdot\gamma\cdot N_{SP}}{A}}{\frac{B}{A}} \tag{51}$$

A and B cancel out to leave the following.

$$R_{NOR} = \gamma\cdot R_{INV}\cdot N_{SP} \tag{52}$$

Calibration For Slow Inputs

Gates are not always driven by a fast input voltage. The slew rate of the voltage depends on the driving gate. Sometimes a really slow gate is the driver, like an AOI333, as shown in Figure 21. To see how slow this gate goes, a chain of AOI333's is set up similar to the chain of inverters. The AOI333's are connected to have the worst-case slew rate. This can be seen in Figure 23.

Like the chain of inverters, the NMOS transistor width is set to twice the minimum width and the PMOS transistor is set to four times the minimum width. The rise and fall times at the output of the sixth AOI333 are measured. The maximum of these two times is the

slowest practical rise and fall time a gate will see at its input. This slow input pulse can cause A and R to change dramatically.

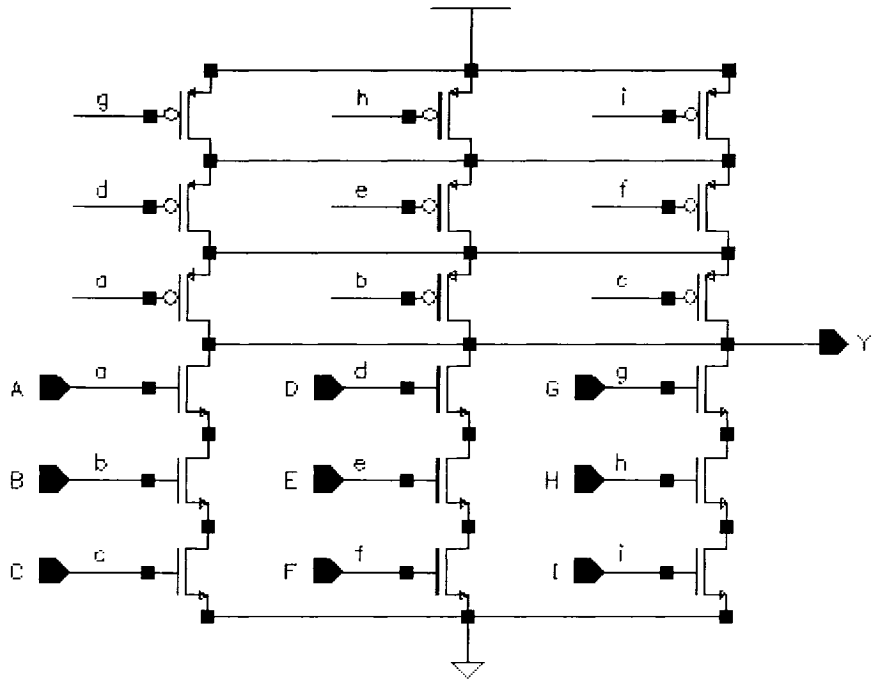

Figure 21. Schematic of an AOI333.

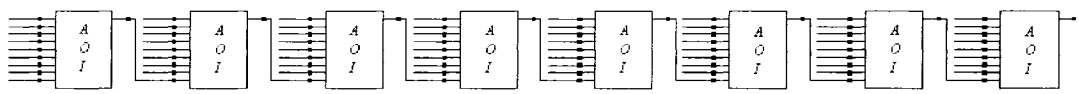

Figure 22. Chain of AOI333's used to find slowest practical slew rate.

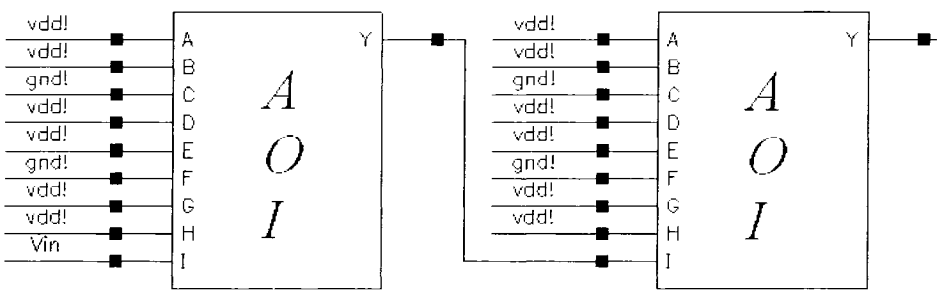

Figure 23. View of AOI333 chain showing inputs and outputs.

To see how A and R change with an increase in rise and fall time, the inverter can be calibrated at this input rise and fall time. The difference is that it is unknown how the change in input changes the propagation delay. Instead of having a delay constraint, a W_N constraint is used. W_N is kept constant from our previous inverter calibration. This makes the "tweaking" of W_P a little different because we no longer have a propagation delay that we are targeting. We will use the following equation for readjusting.

$$W_P = W_P' \cdot \frac{\tau_{PLH}}{\tau_{PHL}} \tag{53}$$

With some process technologies, this equation might over correct W_P. An over correction would be if W_P were initially too big, but then become too small after tweaking. In that case, it would be known what value for W_P is too big and what value is too small. The correct W_P would be somewhere in between. You could then use your own judgment to adjust W_P.

The inverter for the AMI 0.6μm process is shown as an example. An initial W_N of 6.63μm and W_P of 12.89μm are used as the initial estimate. These are the transistor sizes calibrated for the inverter with a fast input. Inserting these values into SPICE and simulating give

$$\frac{\tau_{PLH}}{\tau_{PHL}} = \frac{252.1ps}{117ps} = 2.1547.$$

This shows that the low to high delay is too slow, meaning W_P is too small. Using (53), W_P calculates to be 27.79μm. This value is inserted into SPICE and simulated.

$$\frac{\tau_{PLH}}{\tau_{PHL}} = \frac{54.8ps}{298.8ps} = 0.183$$

Now W_P is too big because the low to high delay is too fast. If (53) is used to readjust, W_P would be 5.1μm. This is obviously a bad estimate because it is already known that 12.98μm is too small, but now it is known that W_P is somewhere between 12.89μm and 27.79μm. The average of the two is taken, 20.34μm, and it is again simulated.

$$\frac{\tau_{PLH}}{\tau_{PHL}} = \frac{130.6ps}{224.6ps} = 0.5822$$

W_P is still too big, but now it can be seen that it lies between 12.89μm and 20.34μm. The average of these two sizes is taken and simulated.

$$\frac{\tau_{PLH}}{\tau_{PHL}} = \frac{182.9ps}{178.7ps} = 1.0235$$

This is really close, but not within one percent yet. It is now known that W_P is closer to 16.62μm than 20.34μm. 17μm is tried next.

$$\frac{\tau_{PLH}}{\tau_{PHL}} = \frac{177.2ps}{183.1ps} = 0.9678$$

Now W_P is too big. 16.7μm is now tried.

$$\frac{\tau_{PLH}}{\tau_{PHL}} = \frac{181.7ps}{176.7ps} = 1.0283$$

W_P is adjusted just a little bit to 16.8μm.

$$\frac{\tau_{PLH}}{\tau_{PHL}} = \frac{180.1ps}{179.7ps} = 1.0022$$

16.8μm yields low to high and high to low delays that are within one percent of each other. A and R can now be calculated as before.

With the inverter calibrated for a slow input pulse voltage, NAND gates can be calibrated in a similar fashion as before. With a slow rising input, however, the input never gets to V_{DD} before the output drops to $V_{DD}/2$. In fact, the input spends most of the time at low voltages. Because of this, V_G is set to 60% of V_{DD}. Also, the output starts dropping before the input gets to $V_{DD}/2$. In order to emulate the NMOS stack, V_D must be swept from not from $V_{DD}/2$ to V_{DD}, but from $V_{DD}/2$ to some voltage lower than V_{DD}. This voltage can be found by looking at the transient waveform as in Figure 24.

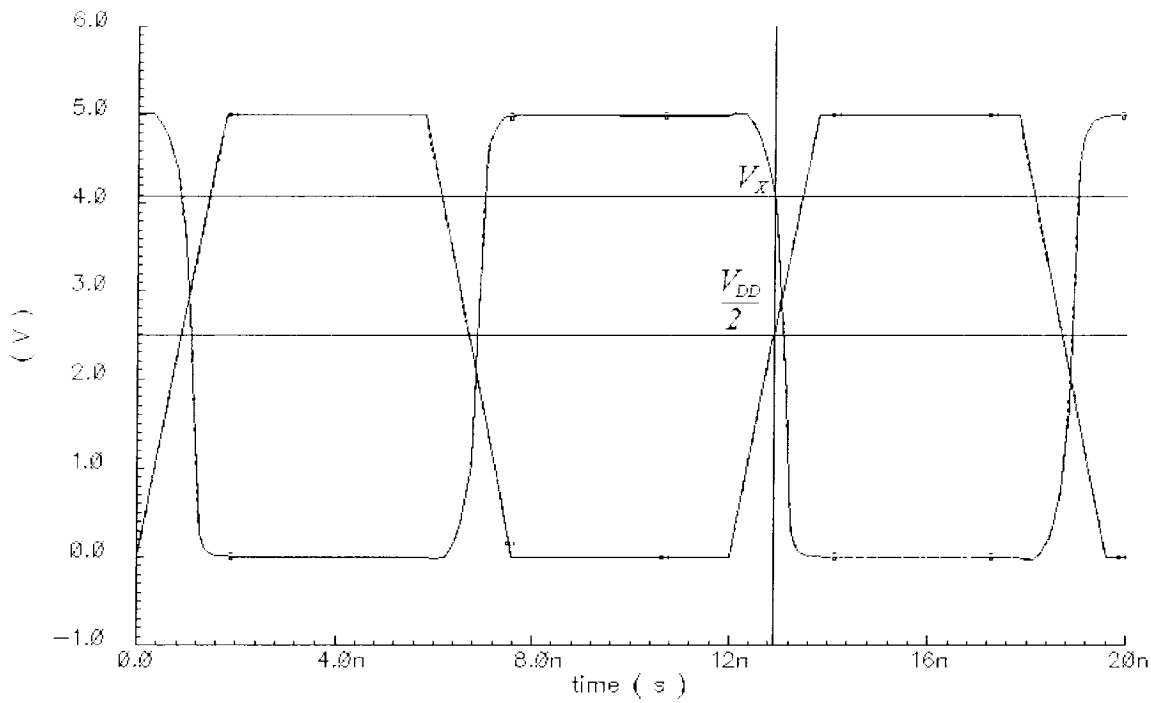

Figure 24. Transient waveform of an inverter with a slow input showing V_X.

In this case, the voltage is 4.1V, but in general, it will be called V_X. We can now find the resistances of the NMOS stacks as before.

$$RES(N_{SN}) = \int_{\frac{V_{DD}}{2}}^{V_X} \frac{1}{I(N_{SN})} dV_D \tag{54}$$

γ, A, and R are calculated the same as before, but with the new resistances and the A and R of the inverter calibrated with the slow input.

Similarly, the PMOS stack of NOR gates can be emulated with a dc simulation as well. This time, V_G is set to 40% of V_{DD} to emulate the slow falling input, and V_D is swept from some voltage, V_Y, to $V_{DD}/2$. V_Y can be found in a similar fashion as V_X. An example is shown in Figure 25.

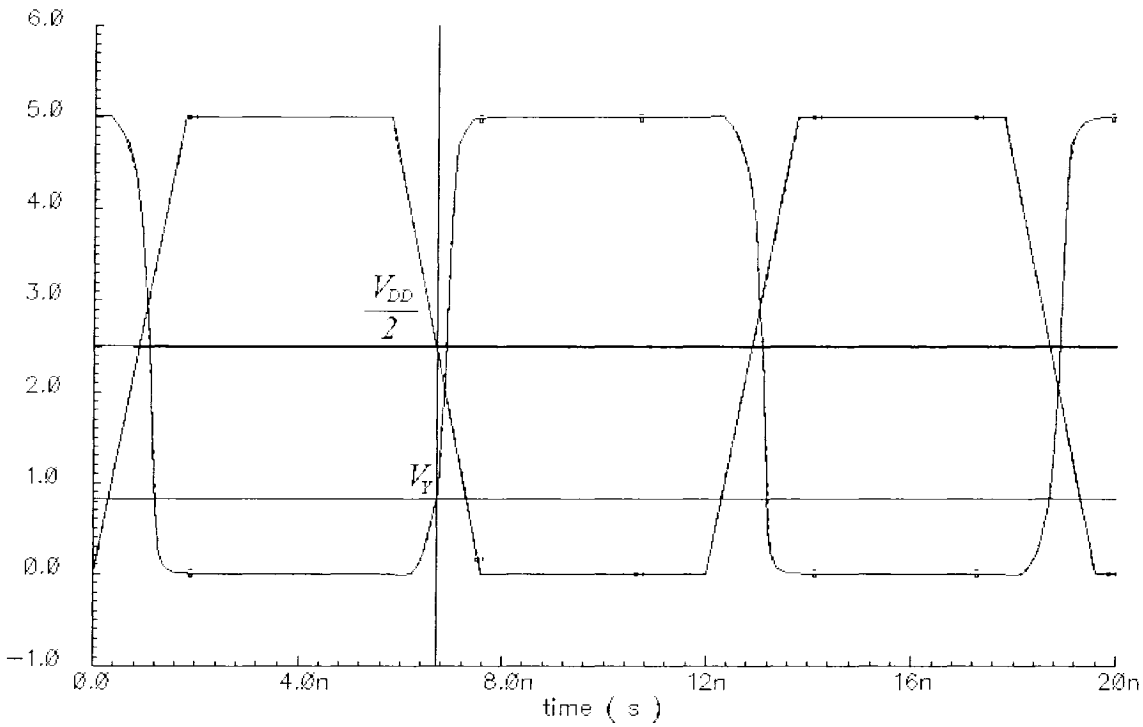

Figure 25. Transient waveform of an inverter with a slow input showing V_Y.

Again, resistance can be calculated as

$$RES(N_{SP}) = \int_{V_Y}^{V_{DD}/2} \frac{1}{I(N_{SP})} dVD \,. \tag{55}$$

γ and R can be calculated the same as before but with the R calibrated for a slow input and the new resistances.

Interpolating Between Fast and Slow

Now that the A and R are calibrated at the two extremes of input slew rate, fast and slow, slew rates in between can be found with a linear fit between the two.

$$A_X = \frac{(A_{slow} - A_{fast})}{(t_{slow} - t_{fast})} \cdot (t_X - t_{fast}) + A_{fast} \tag{56}$$

$$R_X = \frac{(R_{slow} - R_{fast})}{(t_{slow} - t_{fast})} \cdot (t_X - t_{fast}) + R_{fast} \tag{57}$$

Where t_{slow} is the slow input rise and fall time from the AOI333 chain. t_{fast} is the fast input rise and fall time from the inverter chain. t_X is the input rise and fall time of interest. A_{fast} and R_{fast} are the A and R calibrated with the fast input. A_{slow} and R_{slow} are the A and R calibrated with the slow input. A_X and R_X are the A and R with an input rise and fall time of t_X.

Verification and Results

Verification Method

In the last chapter, a method of calibrating the inverter, NANDs, and NORs for a wide range of input slew rates is presented. This chapter deals with testing and verifying that method. The reason for using this method is to find transistor dimensions of a gate with a specific propagation delay constraint. It is already evident that this method speeds up the calibration process, but how accurate is it? When a gate is designed using this method, how close is the simulated delay compared to the targeted delay? The only way to answer these questions is to use the method to size the gates, simulate them in SPICE, and measure the error between the simulated propagation delay and the targeted propagation delay. All of these designs and simulations can be a lengthy process. It would be a very time consuming task to verify every gate, with every load, every input slew rate, and every propagation delay possible. For this reason, testing must be constrained to certain cases.

The gates that will be tested are the inverter, NAND2, NAND3, NAND4, NOR2, NOR3, and NOR4. These gates will be tested with FO1 inverter and FO4 inverters loads. A FO1 inverter is a very common load, and a FO4 inverters is normally the largest load driven by a gate. At each load, the gates will be tested with a fast input, slow input, and a slew rate in between. The fast input rise and fall time is taken from the chain of inverters analysis described in the last chapter. The slow input rise and fall time is taken from the chain of AOI333s analysis. The average of these two times will be used to test the slew rate in between fast and slow. The slowest propagation delay occurs normally when

either the PMOS or NMOS transistors are at the minimum width. For a fast delay, there is some point in size when the delay ceases to decrease with any significance. That is normally the maximum size used. That size will be assumed to be fifty times the minimum length. The delays that yield these two sizes using the calibration method will be tested. The error between simulated and targeted delays will be calculated. These errors will approximately be the range of errors that this method produces.

To ensure that a designer does not spend too much time simulating and tweaking a design, less than 20% error for an initial estimate is desired. Does this method estimate sizes that produce less than 20% error in delay? If so, what propagation delay range does it do this for? This propagation delay range is found for each gate, if any.

This calibration method is to be used across CMOS process technologies of various sizes. It will be tested on three different process technologies: AMI 0.6μm, TSMC 0.18μm, and IBM 0.13μm. The transistor SPICE models of these technologies are shown in Appendix A.

The above test shows how accurate circuits can be designed on the gate level using the calibration method. It will also be shown that this method can be used on a larger level with a circuit that consists of many gates. This will be done with a design example.

A 64-bit Kogg-Stone Adder is used for the example. This adder is a very large circuit consisting of many different gates. The Kogg-Stone Adder is a tree adder. It is very fast and is often used for high-performance 32-bit and 64-bit adders. The delay of the circuit, τ_{TREE}, is shown in [50] to be

$$\tau_{TREE} = \tau_{PG} + [\log_2 M] \cdot \tau_{AO} + \tau_{XOR} \quad (58)$$

τ_{PG} is the propagation delay of the propagate and generate gates. The propagate and generate gates are shown in Figures 27 and 28. M is the number of bits of the adder. τ_{AO} is the delay of the AND-OR gate of the black cell. The black cell is shown in Figure 29. τ_{XOR} is the delay of the sum gate, which is an XOR gate. The sum gate is shown in Figure 30.

The architecture of the Kogg-Stone Adder consists of propagate gates, generate gates, black cells, sum gates, and a carry-out gate. The carry-out gate is shown in Figure 31. The architecture of a Kogg-Stone Adder is shown in Figure 26. This design example is a 64-bit adder, but a 16-bit adder is shown here to preserve space and details. *A16-A1* and *B16-B1* are the two binary numbers to be added. *CIN* is the input carry bit. *S16-S1* is the sum of the two input numbers and the input carry. *COUT* is the carry out.

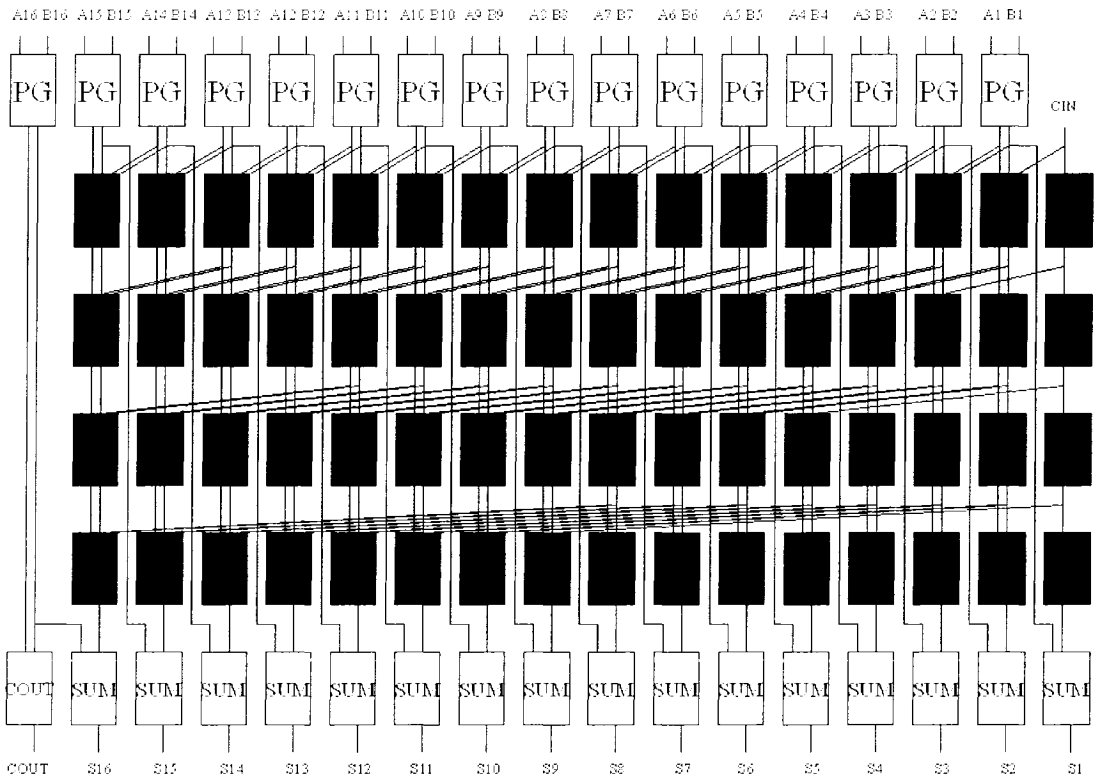

Figure 26. Example of a 16-bit Kogg-Stone Adder.

A 64-bit Kogg-Stone Adder follows the same pattern as the 16-bit version. In the 64-bit version, there are sixty-four propagate gates, generate gates, and sum gates. There are also sixty-four columns and six rows of black cells. The two numbers to be added are *A64-A1* and *B64-B1*.

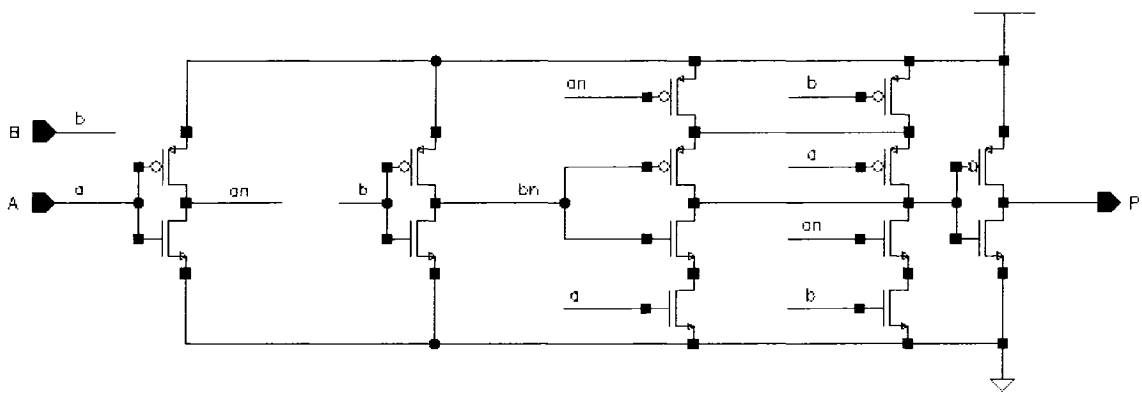

Figure 27. Schematic of propagate gate.

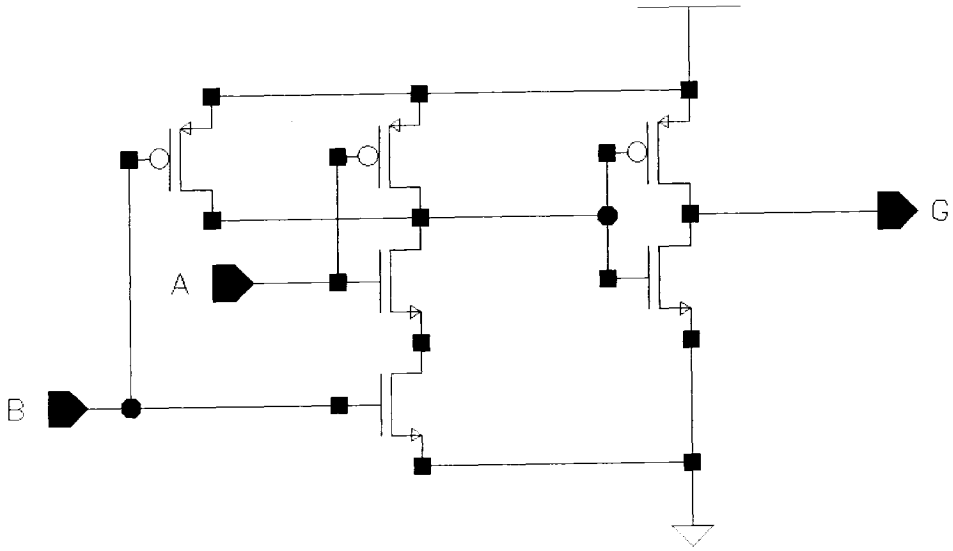

Figure 28. Schematic of generate gate.

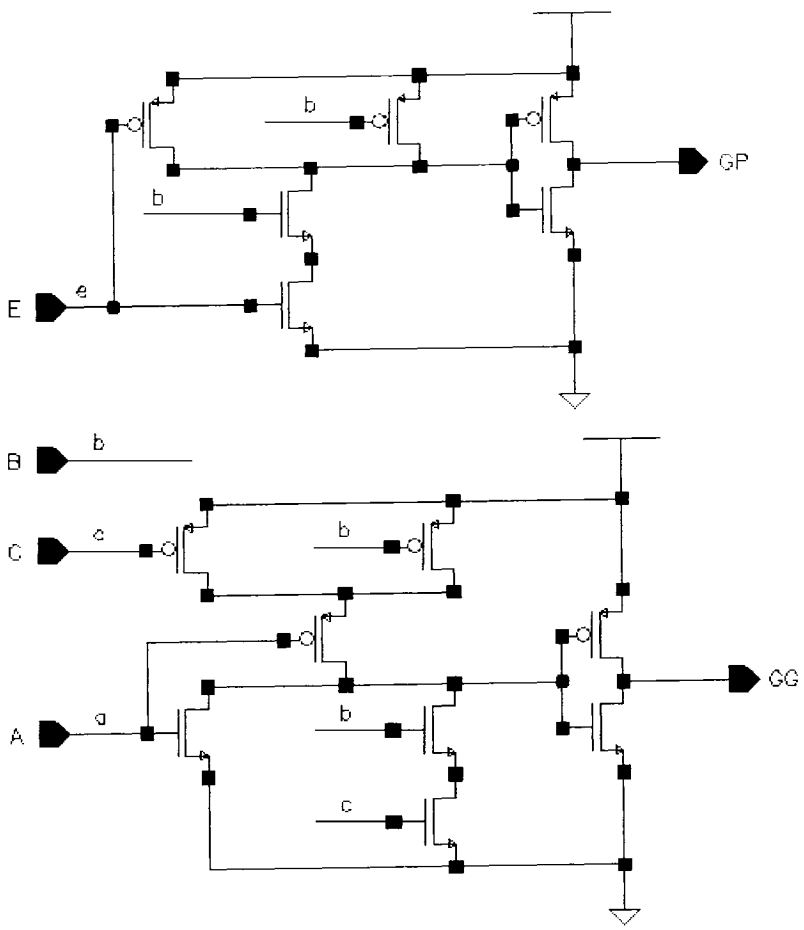

Figure 29. Schematic of black cell.

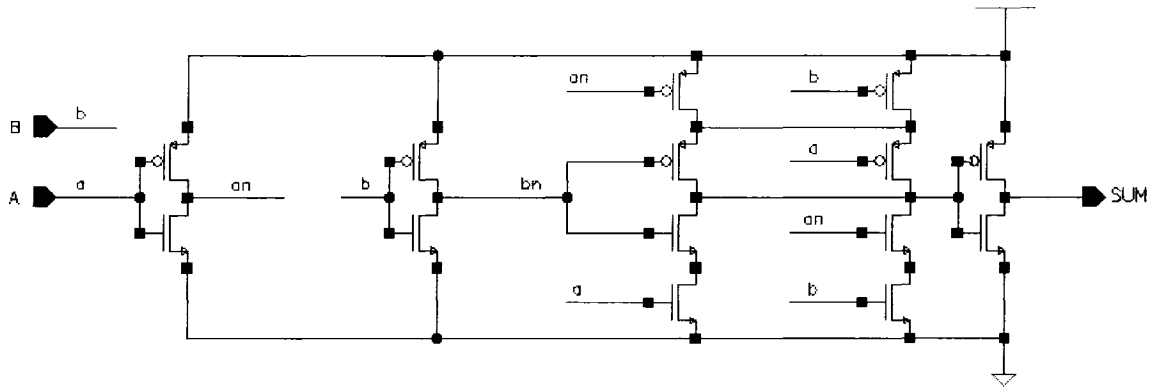

Figure 30. Schematic of sum gate.

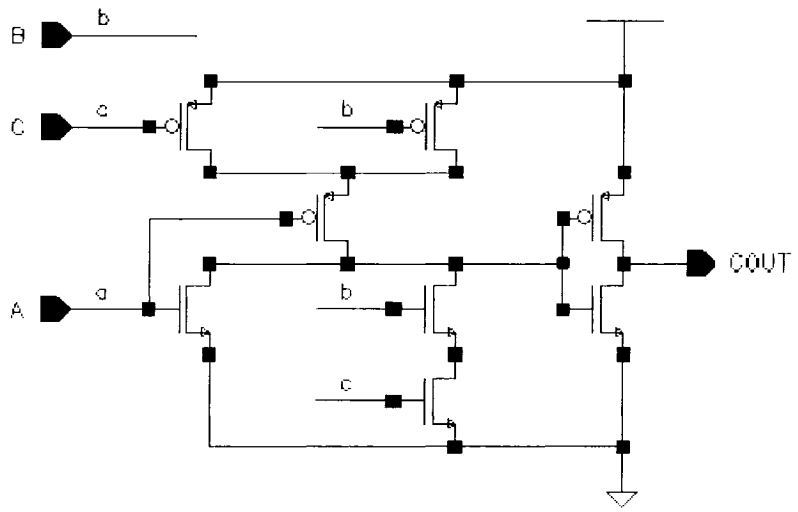

Figure 31. Schematic of carry-out gate.

This design example is done in the TSMC 0.18μm process technology. To size the transistors, timing specifications are given for each gate. For the propagate gate, 50ps is targeted for the output inverter, 100ps for the XNOR gate, and 50ps for the input inverters. For the generate gate, 100ps is targeted for the inverter, and 100ps is targeted for the NAND gate. The AND-OR gate of the black cell is targeted to have a propagation delay of 100ps for the inverter and 100ps for the AND-OR portion of the gate. The inverter of the AND portion of the black cell is the same one used for the AND-OR gate. The NAND gate of the AND portion of the black cell is targeted to have 100ps delay.

The sum block is targeted to have 100ps for the XNOR gate and 50ps for all of the inverters. The carry-out gate uses the same inverter as the output inverter of the sum gate, and 100ps is targeted for the AND-OR portion of it. Two times the propagation delay of the driving gate is used as the input rise and fall time.

Some things that need to be considered in this design are the AND-OR and XNOR gates and the load capacitance estimations. In the previous chapter, the method was only applied to inverters, NANDs, and NORs. Now we have AND-ORs and XNORs to design. How are A and R estimated for these gates? The AOI, in this case, has one NMOS in parallel with a stack of two NMOS transistors. If the worst-case delay is considered, the bottom NMOS of the two-stack is switching. That means that the A is calibrated the same way that A of a NAND2 is calibrated. The AND-OR also has a two-stack of PMOS transistors. That means that the B can be calibrated the same way as the B of a NOR gate. This leaves R to be calibrated as

$$R = \frac{RES(N_{SP}) \cdot RES(N_{SN} = 1)}{RES(N_{SP} = 1) \cdot RES(N_{SN})} \cdot R_{INT}. \tag{59}$$

Since the XNOR also has a stack of two NMOS and a stack of two PMOS transistors, the A and R can be calibrated the same way as the AND-OR.

The load capacitance is estimated to be the gate capacitances of the transistors that are being driven. For instance, the generate gate is driving the AND-ORs of two black cells. In this case, the load capacitance will be

$$C_G = 2 \cdot (W_N \cdot L_N \cdot C_{GBN} + W_P \cdot L_P \cdot C_{GBP} + W_N \cdot C_{GDON} + W_P \cdot C_{GDOP}). \tag{60}$$

Where the W_N and W_P in this case are the widths of the black cell transistors. The load capacitance of the other gates can be estimated in the same fashion except for the black cells. The black cell is a special case. The worst case is what is being designed for. For black cells, the worst-case load capacitance is when the black cells drive two of themselves plus some interconnect capacitance. The interconnect capacitance, in this case, cannot be neglected because the wires are very long. The interconnect capacitance can easily be estimated as will be shown, but the gate capacitance cannot be. If the gate drives itself, and the capacitance depends on the size of the gate, but the sizes depend on the load capacitance, then this is an infinite loop. To get around this loop, the capacitance of the carry-out gate is used instead. The carry-out gate is an AND-OR also, so this is the closest estimate to the AND-OR capacitance of the black cell.

The wire capacitance that is of concern is of the long wire that is to stretch across thirty-three black cells laid out side by side. If we take the width of a NAND4 to be the worst case width of the layout of a cell, then that width can be multiplied by thirty-three to give the length of the wire. The width of a NAND4 is estimated to be 6.03μm. The wire not only stretches across thirty-two black cells length-wise, but it also stretches across two black cells height-wise and some space between the two black cells. The height of a black cell is estimated to be 20.07μm and the space between the two black cells is estimated to be 52.11μm. Metal one is used for the length-wise wire, and metal two is used for the height-wise wire. The metal one width is 0.54μm, and the metal two width is 0.36μm. This gives a metal one area of 107.455μm^2, a metal one perimeter of 399.06μm, a metal two area of 33.21μm^2, and a metal two perimeter of 185.22μm. Area

capacitance and fringe capacitance numbers for metal one and metal two are taken from a MOSIS fabrication run. Metal one area capacitance is 39aF/μm^2, metal one fringe capacitance is 0af/μm, metal two area capacitance is 18aF/μm^2, and metal two fringe capacitance is 59aF/μm. This gives a total interconnect capacitance of 15.7165fF.

The other load capacitance that is unknown is that at the output of the sum and carry-out gates. A FO4 inverters load is used in this case.

To show an example of how the calibration method is used to design this circuit, the inverter and XNOR of the sum block are now shown step-by-step. The inverter of the sum block is the first gate sized in this design because all other gates depend on the size of this gate. Using the calibration method, it is seen that the A and R calibrated for a 40ps input rise and fall time are 12539Ω and 2.2657. The A and R calibrated for a 600ps input rise and fall time are 26048Ω and 3.1336. The A and R at 40ps is calibrated the "old" way as described in the Literature Review of this thesis. The A and R calibrated at 600ps is calibrated with W_N kept constant as described in the Method chapter of this thesis. This gate is not to be designed with a 40ps or a 600ps rise and fall time but a 200ps input rise and fall time. This input voltage comes from the output of the driving gate, the XNOR of the sum block. The XNOR is going to be designed to have a 100ps propagation delay. As mentioned above, the rise and fall time is estimated as twice the propagation delay of the driving gate. Using (56) and (57), A and R can be calculated for a 200ps input rise and fall time, 16399Ω and 2.5137. As stated above, the capacitance loading the inverter is a FO4, 14.27fF. The target propagation delay for this gate is 50ps. With A, R, load capacitance, and target delay known, W_N and W_P can be calculated using

(36) and (37). N and M, in this case are both 1. W_N is calculated to be 1.662μm, and W_P is calculated to be 4.177μm. The input capacitance, with these sizes, is 12.86fF. This is the load capacitance of the XNOR gate. The XNOR gate has an inverter with a 50ps delay driving it. This means that the input is of the XNOR gate is estimated to have a 100ps input rise and fall time. Using the calibration method, A and R are estimated to be 9268Ω and 2.818. The A in this case is calibrated the same way the A of a NAND2 is calibrated. The R is calibrated using (59). Again, (56) and (57) are used to interpolate between fast and slow inputs. The target delay for this gate is 100ps. Using these numbers along with (36) and (37), W_N is calculated to be 3.824μm, and W_P is calculated to be 10.776μm. The weighted number of drains, N and M, are both 4 in this case. This process is repeated with the rest of the gates.

The sizes can be seen in Table 4. No readjusting is done. Only the calculated sizes are inserted into schematic, and a SPICE simulation is run to measure the propagation delay of the adder. With this, it can be seen how accurately the calibration method can be used to design large circuits.

Table 4. Transistor sizes of Kogg-Stone Adder.

Cell	$C_G + C_{INT}$ F	τ_P s	t_{IN} ps	A Ω	R none	N	M	N_{SN}	N_{SP}	W_N cm	W_P cm
INV_{sum_out}	1.427E-14	5E-11	200	16399	2.513672	1	1	1	1	1.6618E-04	4.1771E-04
$XNOR_{sum}$	1.286E-14	1E-10	100	9268	2.818042	4	4	2	2	3.8239E-04	1.0776E-03
INV_{sum_in}	3.215E-14	5E-11	200	16399	2.513672	1	1	1	1	3.6964E-04	9.2915E-04
AOI_{carry_out}	1.286E-14	1E-10	200	10292	2.898963	3	2.5	2	2	1.5428E-04	4.4725E-04
INV_{black}	5.43E-14	1E-10	200	16399	2.513672	1	1	1	1	2.1233E-04	5.3373E-04
AOI_{black}	1.643E-14	1E-10	200	10292	2.898963	3	2.5	2	2	1.9430E-04	5.6326E-04
$NAND2_{black}$	1.643E-14	1E-10	200	10292	1.982933	2	2	2	1	1.0984E-04	2.1781E-04
$INV_{generate}$	1.668E-14	1E-10	200	16399	2.513672	1	1	1	1	6.6135E-05	1.6624E-04
$NAND2_{generate}$	5.118E-15	1E-10	200	10292	1.982933	2	2	2	1	3.7140E-05	7.3646E-05
$INV_{propagate\_out}$	3.112E-14	1E-10	200	16399	2.513672	1	1	1	1	1.2224E-04	3.0727E-04
$XNOR_{propagate}$	9.46E-15	1E-10	100	9268	2.818042	4	4	2	2	2.9072E-04	8.1925E-04
$INV_{propagate\_in}$	2.445E-14	5E-11	200	16399	2.513672	1	1	1	1	2.8193E-04	7.0867E-04

Results

After testing the method on the IBM 0.13μm process, it can be seen that the worst positive error occurs with the inverter with a FO1 load, an input rise and fall time of 222ps, and a W_P of fifty times the minimum gate length, L_{MIN}. This case yields a +60.2% error meaning the measured delay is 60.2% greater than the targeted delay. The worst negative error occurs with the NAND4 with a FO4 load, a 222ps input rise and fall time, and minimum width for W_N. This case gives a –41.5% error meaning the measured delay is less than the targeted delay by 41.5%. There are a few cases in which the error never gets within 20%. This happens with the inverter with a FO1 load and a 222ps input rise and fall time and the NOR4 with a FO1 load and a 410ps input rise and fall time. All other cases have a range of propagation delays that are predicted within 20% error. These ranges are presented in Table 5 in Appendix B. There are also cases in which the propagation delays are accurate within 20% error for the full range of transistor widths

tested, W_{MIN} to $50 \cdot L_{MIN}$. These cases are the NAND2 with a FO1 load and a fast input, the NAND3 with a FO1 load and a fast and slow input, the NAND4 with a FO1 load and the whole range of inputs, the NOR2 and NOR3 with a FO4 and fast inputs, a NOR4 with a FO1 load and a 222ps input, and the NOR4 with a FO4, a fast input, and a 222ps input.

The worst positive error for the TSMC 0.18μm process is +37.9%. This occurs with the inverter with a FO1 load, a 320ps input, and a W_P equal to fifty times L_{MIN}. The worst negative error is –49.5%. This occurs with the NOR4 with a FO4 load, a 600ps input, and a W_N at minimum width. All cases are accurate within 20% error for some range of propagation delays. This range is shown in Figure 39 in Appendix B. Unlike the IBM 0.13μm process, there are no cases that work within 20% error for the full range of transistor sizes.

The worst positive error for the AMI 0.6μm process is +51.7%. This occurs with the inverter with a FO1 load, a 963ps input, and a W_P sized at fifty times L_{MIN}. The worst negative error is –59.4%. This happens with a NOR4, an 1800ps input, and a minimum W_N. There are several cases in which the error never drops below 20% no matter what the propagation delay is. These cases were the NAND3, NAND4, NOR3, and NOR4 with slow inputs. The NOR4 with a FO1 and a fast input also never drops below 20% error. On the other hand, there are cases in which the full range of transistor widths fits within 20% error. These cases are the inverter and the NANDs with a fast input. The NOR2 with a FO4 load and a fast input also fits within 20% error for the full range of transistor widths.

After designing the adder and inserting it into a schematic, it was simulated. The following combinations of the input were tested.

All A's=CIN=0 with all B's fluctuating

All A's fluctuating with All B's=CIN=0

All A's=1, All B's=0, and CIN fluctuating

All A's=1, $B1$ fluctuating, and all other B's=CIN=0

The worst-case delay occurred with all A inputs tied high, $B1$ fluctuating, all other B inputs tied low, and CIN tied low. The output, $S64$, provides the slowest delay. The delay came out to be 1.4341ns for the low to high delay and 1.4451ns for the high to low delay. Knowing that the worst-case delay is calculated from (58) to be 1.65ns, the error came out to be –13.1% and –12.4%, respectively.

Analysis

In this thesis, a method for calibrating the Kang and Leblebici propagation delay model for various digital gates is presented. This method requires no more than twenty-four simulations to calibrate and inverter and various other gates such as NAND and NOR gates. Approximately twenty simulations are needed to calibrate the inverter, and only four more simulations are needed to calibrate all other gates. This calibration method takes into account input slew rate variations and carrier velocity saturation. Because this method takes into account input slew rate, it allows each gate in a circuit to be designed in parallel. This is because the sizes can be estimated fairly accurately, allowing a designer to estimate the load capacitance fairly accurately without needing the loading gates to be designed yet. The design example is an example of this. The design example also shows that the propagation delay of a large circuit can be estimated using this method. This would allow a circuit designer to estimate how fast a circuit architecture can run and to make needed tradeoffs.

This method, along with the Kang and Leblebici delay model, rivals the Logical Effort model. Both of these methods can be used to estimate the speed of a circuit architecture. However, Logical Effort does not show a relationship between propagation delay and transistor sizes. Logical Effort assumes all transistor sizes to be the same as those of the standard inverter. This method presented in this thesis allows a designer to make tradeoffs not only with circuit architectures, but also with transistor sizes. In order for Logical Effort to be more accurate, each gate must be calibrated, requiring several

simulations per gate. The method of this thesis requires only four simulations to calibrate gates other than the inverter.

This method can be inaccurate for some cases. An error of up to 60% has been seen. The inverter, even though calibrated with the most accuracy, yields this 60% error. The inverter is calibrated with a fast input and a slow input. The 60% error occurs at the average input slew rate. This may be because A and R may not be linear with respect to input slew rate. In this case, a more accurate way of interpolating between fast and slow inputs may be needed. Other cases of high error are for NANDs and NORs with slow inputs. This may be due to the fact that the transistors do not turn off instantaneously. This factor is neglected in the calibration dc simulations.

Bibliography

[1] L. W. Nagel, "The Life of SPICE," in *Bipolar Circuits and Technology Meeting*, Minneapolis, MN, 1996, pp. 1-4.

[2] I. Sutherland, B. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*. San Francisco, CA: Morgan Kaufmann Publishers, 1999.

[3] S. M. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits*. New York, NY: McGraw-Hill, 2003.

[4] T. Sakurai and A. R. Newton, "Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas," *IEEE Journal of Solid-State Circuits*, vol. 25, pp. 584-594, Apr. 1990.

[5] S. R. Vemuru and A. R. Thorbjornsen, "A CMOS Inverter Model for Propagation Delay Evaluation," in *Proc. 32nd Midwest Symposium Circuits and Systems*, Champaign, IL, 1989, pp. 563-566.

[6] L. Bisdounis, S. Nikolaidis, O. Koufopavlou, and C. Goutis, "Accurate Timing Model for the CMOS Inverter," in *Proc. 3rd IEEE Int. Conf. Electronics, Circuits, and Systems*, Rodos, 1996, pp. 89-92.

[7] J. L. Rossello and J. Segura, "An Analytical Charge-Based Compact Delay Model for Submicrometer CMOS Inverters," *IEEE Trans. Circuits and Systems*, vol. 51, pp. 1301-1311, July 2004.

[8] A. A. Hamoui and N. C. Rumin, "An Analytical Model for Current, Delay, and Power Analysis of Submicron CMOS Logic Circuits," *IEEE Trans. Circuits and Systems*, vol. 47, pp. 999-1007, Oct. 2000.

[9] L. Bisdounis, S. Nikolaidis, O. Koufopavlou, and C. Goutis, "Analytical Transient Response and Propagation Delay Evaluation of the CMOS Inverter for Short-Channel Devices," *IEEE Journal of Solid-State Circuits*, vol. 33, pp. 302-306, Feb. 1998.

[10] S. Nikolaidis, A. Chatzigeorgiou, and E. D. Kyriakis-Bitzaros, "Delay and Power Estimation for a CMOS Inverter Driving RC Interconnect Loads," in *Proc. 1998 IEEE Int. Symposium Circuits and Systems*, Monterey, CA, 1998, pp. 368-371.

[11] K. T. Tang and E. G. Friedman, "Delay and Power Expressions Characterizing a CMOS Inverter Driving an RLC Load," in *Proc. 2000 IEEE Int. Symposium Circuits and Systems*, Geneva, 2000, pp. 283-286.

[12] V. Adler and E. G. Friedman, "Delay and Power Expressions for a CMOS Inverter Driving a Resistive-Capacitive Load," in *Proc. 1996 IEEE Int. Symposium Circuits and Systems*, Atlanta, GA, 1996, pp. 101-104.

[13] D. Burdia, G. Grigore, and C. Ionascu, "Delay and Short-Circuit Power Expressions Characterizing a CMOS Inverter Driving Resistive Interconnect," in *Int. Symposium Signals, Circuits, and Systems*, 2003, pp. 597-600.

[14] S. H. K. Embabi and R. Damodaran, "Delay Models for CMOS, BiCMOS and BiNMOS Circuits and Their Applications for Timing Simulations," *IEEE Trans. Computer-Aided Design Integrated Circuits and Systems*, vol. 13, pp. 1132-1142, Sept. 1994.

[15] K. Chen, C. Hu, P. Fang, and A. Gupta, "Experimental Confirmation of an Accurate CMOS Gate Delay Model for Gate Oxide and Voltage Scaling," *IEEE Electron Device Letters*, vol. 18, pp. 275-277, June 1997.

[16] P. Maurine, N. Azemard, and D. Auvergne, "General Representation of CMOS Structure Transition Time for Timing Library Representation," *Electronics Letters*, vol. 38, pp. 175-177, Feb. 2002.

[17] H. C. Chow and W. S. Feng, "Model for Propagation Delay Evaluation of CMOS Inverter Including Input Slope Effects For Timing Verification," *Electronics Letters*, vol. 28, pp. 1159-1160, June 1992.

[18] A. Chatzigeorgiou, S. Nikolaidis, and I. Tsoukalas, "Modeling CMOS Gates Driving RC Interconnect Loads," *IEEE Trans. Circuits and Systems*, vol. 48, pp. 413-418, Apr. 2001.

[19] K. Jeppson, "Modeling the Influence of the Transistor Gain Ratio and the Input-to-Output Coupling Capacitance on the CMOS Inverter Delay," *IEEE Journal of Solid-State Circuits*, vol. 29, pp. 646-654, June 1994.

[20] L. Bisdounis, O. Koufopavlou, and S. Nikolaidis, "Modelling Output Waveform and Propagation Delay of a CMOS Inverter in the Submicron Range," *IEE Proc. Circuits, Devices and Systems*, vol. 145, pp. 402-408, Dec. 1998.

[21] Y. I. Ismail and E. G. Friedman, "Optimum Repeater Insertion Based on a CMOS Delay Model for On-Chip RLC Interconnect," in *Proc. 11th Annual IEEE Int. ASIC Conf.*, Rochester, NY, 1998, pp. 369-373.

[22] L. Bisdounis, S. Nikolaidis, and O. Koufopavlou, "Propagation Delay and Short-Circuit Power Dissipation Modeling of the CMOS Inverter," *IEEE Trans. Circuits and Systems*, vol. 45, pp. 259-270, Mar. 1998.

[23] D. Auvergne, J. M. Daga, and M. Rezzoug, "Signal Transition Time Effect on CMOS Delay Evaluation," *IEEE Trans. Circuits and Systems*, vol. 47, pp. 1362-1369, Sept. 2000.

[24] L. Bisdounis, S. Nikolaidis, O. Koufopavlou, and C. Goutis, "Switching Response Modeling of the CMOS Inverter for Sub-micron Devices," in *Proc. Design, Automation and Test in Europe*, Paris, 1998, pp. 729-735.

[25] K. T. Tang and E. G. Friedman, "Transient Analysis of a CMOS Inverter Driving Resistive Interconnect," in *Proc. 2000 IEEE Int. Symposium Circuits and Systems*, Geneva, 2000, pp. 269-272.

[26] H. Kriplani, F. Najm, and I. Hajj, "Improved Delay and Current Models for Estimating Maximum Currents in CMOS VLSI Circuits," in *1994 IEEE Int. Symposium Circuits and Systems*, London, 1994, pp. 435-438.

[27] S. M. Kang, "A Design of CMOS Polycells for LSI Circuits," *IEEE Trans. Circuits and Systems*, vol. CAS-28, pp. 838-843, Aug. 1981.

[28] M. Bafleur, J. Buxo, J. P. Teixeira, and I. C. Teixeira, "A Logical Timing Simulator for CMOS Circuits Based on an Accurate Formulation of the Propagation Delay," in *European Conf. Circuit Theory and Design*, Brighton, 1989, pp. 270-274.

[29] Y. H. Yang and C. Y. Wu, "Analysis and Modelling of Initial Delay Time and Its Impact on Propagation Delay of CMOS Logic Gates," *IEE Proc. Circuits, Devices and Systems*, vol. 136, pp. 245-254, Oct. 1989.

[30] N. Hedenstierna and K. O. Jeppson, "CMOS Circuit Speed and Buffer Optimization," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 270-281, Mar. 1987.

[31] D. Etiemble, V. Adeline, N. H. Duyet, and J. C. Ballegeer, "Micro-Computer Oriented Algorithms for Delay Evaluation of MOS Gates," in *21st Conf. Design Automation*, 1984, pp. 358-364.

[32] J. C. Andre, J. P. Teixeira, I. C. Teixeira, J. Buxo, and M. Bafleur, "Propagation Delay Modelling of MOS Digital Networks," in *Proc. Electrotechnical Conf.*, Lisbon, 1989, pp. 311-314.

[33] V. Gerousis, N. Phan, and D. Weaver, "New Delay Model for 0.5μ CMOS ASIC," in *Proc. 6th Annual IEEE Int. ASIC Conf. And Exhibit*, Rochester, NY, 1993, pp. 511-514.

[34] M. Shams and M. I. Elmasry, "Delay Optimization of CMOS Logic Circuits Using Closed-Form Expressions," in *Int. Conf. Computer Design*, Austin, TX, 1999, pp. 563-568.

[35] B. Lasbouygues, S. Engels, R. Wilson, P. Maurine, N. Azemard, and D. Auvergne, "Logical Effort Model Extension to Propagation Delay Representation," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 1677-1684, Sept. 2006.

[36] P. Maurine, M. Rezzoug, N. Azemard, and D. Auvergne, "Transition Time Modeling in Deep Submicron CMOS," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, pp. 1352-1363, Nov. 2002.

[37] J. L. Rossello, C. de Benito, and J. Segura, "A Compact Gate-Level Energy and Delay Model of Dynamic CMOS Gates," *IEEE Trans. Circuits and Systems*, vol. 52, pp. 685-689, Oct. 2005.

[38] J. L. Rossello and J. Segura, "A Compact Propagation Delay Model for Deep-Submicron CMOS Gates Including Crosstalk," in *Proc. Design, Automation and Test in Europe Conf. and Exhibition*, 2004, pp. 954-959.

[39] X. Jiang, A. P. Jayasumana, W. Zhang, and S. Chiao, "A Proper Deep Submicron MOSFET Model (PDSMM) and Its Applications for Delay Modeling of CMOS Inverters," in *Proc. 6th Int. Conf. Solid-State and Integrated-Circuit Technology*, 2001, pp. 875-878.

[40] K. Chen, C. Hu, P. Fang, A. Gupta, M. R. Lin, and D. L. Wollesen, "Accurate Models for CMOS Scaling and Gate Delay in Deep Sub-micron Regime," in *1997 Int. Conf. Simulation of Semiconductor Processes and Devices*, Cambridge, MA, 1997, pp. 261-264.

[41] C. Y. Wu, J. S. Hwang, C. Chang, and C. C. Chang, "An Efficient Timing Model for CMOS Combinational Logic Gates," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 636-650, Oct. 1985.

[42] B. Lasbouygues, J. Schindler, S. Engels, P. Maurine, N. Azemard, and D. Auvergne, "Continuous Representation of the Performance of a CMOS Library," in *Proc. 29th European Solid-State Circuits Conf.*, 2003, pp. 595-598.

[43] S. R. Vemuru and A. R. Thorbjornsen, "Delay-Modeling of NAND Gates," in *Proc. 33rd Midwest Symposium Circuits and Systems*, Calgary, Alta., 1990, pp. 922-925.

[44] A. Hirata, H. Onodera, and K. Tamaru, "Estimation of Propagation Delay Considering Short-Circuit Current for Static CMOS Gates," *IEEE Trans. Circuits and Systems*, vol. 45, pp. 1194-1198, Nov. 1998.

[45] S. Nikolaidis and A. Chatzigeorgiou, "Modeling the Transistor Chain Operation in CMOS Gates for Short Channel Devices," *IEEE Trans. Circuits and Systems*, vol. 46, pp. 1191-1202, Oct. 1999.

[46] P. Maurine, M. Rezzoug, and D. Auvergne, "Output Transition Time Modeling of CMOS Structures," in *2001 IEEE Int. Symposium Circuits and Systems*, Sydney, NSW, 2001, pp. 363-366.

[47] J. L. Rossello and J. Segura, "Power-delay Modeling of Dynamic CMOS Gates for Circuit Optimization," in *IEEE/ACM Int. Conf. Computer Aided Design*, San Jose, CA, 2001, pp. 494-499.

[48] J. L. Rossello and J. Segura, "Simple and Accurate Propagation Delay Model for Submicron CMOS Gates Based on Charge Analysis," *Electronics Letters*, vol. 38, pp. 772-774, July 2002.

[49] R. J. Baker, *CMOS Circuit Design, Layout, and Simulation*, 2nd Ed. Piscataway, NJ: IEEE Press, 2005.

[50] N. H. E. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 3rd Ed. Boston, MA: Pearson Education, Inc., 2005.

Appendix A: SPICE Models

The following SPICE models were used in the simulations for calibration and verification. Figures 32 and 33 are the SPICE models of NMOS and PMOS transistors, respectively, used for the IBM 0.13μm process. Figures 34 and 35 are the transistor models for the TSMC 0.18μm process. Figures 36 and 37 are the transistor models for the AMI 0.6μm process.

```
.MODEL ibm13dN NMOS( LEVEL=11
VERSION=3.1          TNOM=27              TOX=3.2E-9
XJ=1E-7              NCH=2.3549E17        VTH0=0.0568751
K1=0.3020081         K2=-0.0126353        K3=1E-3
K3B=4.0207582        W0=1E-7              NLX=1E-6
DVT0W=0              DVT1W=0              DVT2W=0
DVT0=1.0831955       DVT1=0.1843614       DVT2=0.274803
U0=434.7091169       UA=-5.43709E-10      UB=3.701463E-18
UC=4.393831E-10      VSAT=1.628587E4      A0=1.7574452
AGS=0.9073285        B0=6.50197E-6        B1=5E-6
KETA=0.0191837       A1=8.559091E-4       A2=0.3645109
RDSW=150             PRWG=0.3557617       PRWB=0.1071626
WR=1                 WINT=1.783755E-8     LINT=1.034003E-8
DWG=-3.45492E-10     DWB=1.327182E-8      VOFF=-0.0378384
NFACTOR=2. 5         CIT=0                CDSC=2.4E-4
CDSCD=0              CDSCB=0              ETA0=1.001562E-3
ETAB=0.5140021       DSUB=4.098585E-6     PCLM=0.9707588
PDIBLC1=0.9980936    PDIBLC2=0.01         PDIBLCB=-0.1
DROUT=0.999035       PSCBE1=7.973198E10   PSCBE2=5.003614E-10
PVAG=0.49933         DELTA=0.01           RSH=6.9
MOBMOD=1             PRT=0                UTE=-1.5
KT1=-0.11            KT1L=0               KT2=0.022
UA1=4.31E-9          UB1=-7.61E-18        UC1=-5.6E-11
AT=3.3E4             WL=0                 WLN=1
WW=0                 WWN=1                WWL=0
LL=0                 LLN=1                LW=0
LWN=1                LWL=0                CAPMOD=2
XPART=0.5            CGDO=4.04E-10        CGSO=4.04E-10
CGBO=1E-10           CJ=8.395926E-4       PB=0.8452447
MJ=0.5347241         CJSW=2.43093E-10     PBSW=0.8
MJSW=0.2862576       CJSWG=3.3E-10        PBSWG=0.8
MJSWG=0.2862576      CF=0                 PVTH0=2.009264E-4
PRDSW=0              PK2=1.30501E-3       WKETA=-1.227146E-3
LKETA=6.04383E-3     PU0=4.4729531        PUA=1.66833E-11
PUB=0                PVSAT=653.2294237    PETA0=1E-4
PKETA=-0.0204684 )
```

Figure 32. NMOS transistor SPICE model used for the IBM 0.13μm process.

```
.MODEL ibm13dP PMOS ( LEVEL=11
VERSION=3.1          TNOM=27              TOX=3.2E-9
XJ=1E-7              NCH=4.1589E17        VTH0=-0.2184693
K1=0.251043          K2=0.0107986         K3=0.0939299
K3B=14.287779        W0=1E-6              NLX=2.83073E-7
DVT0W=0              DVT1W=0              DVT2W=0
DVT0=8.124641E-4     DVT1=0.4664019       DVT2=-0.1
U0=111.8794007       UA=1.325562E-9       UB=3.221853E-21
UC=-1.73309E-11      VSAT=1.998045E4      A0=2
AGS=0.7381126        B0=6.526983E-6       B1=5E-6
KETA=0.0167193       A1=1.316147E-3       A2=0.4631991
RDSW=106.4080613     PRWG=-0.4993469      PRWB=0.5
WR=1                 WINT=0               LINT=7.66174E-9
DWG=4.097251E-9      DWB=-3.051451E-8     VOFF=-0.1022829
NFACTOR=1.5332272    CIT=0                CDSC=2.4E-4
CDSCD=0              CDSCB=0              ETA0=7.088326E-4
ETAB=-3.405323E-3    DSUB=1.642777E-3     PCLM=0.1688598
PDIBLC1=1.428801E-3  PDIBLC2=-9.415836E-3  PDIBLCB=-1E-3
DROUT=3.682163E-6    PSCBE1=5.118004E9    PSCBE2=1.563874E-9
PVAG=0.0183422       DELTA=0.01           RSH=6.9
MOBMOD=1             PRT=0                UTE=-1.5
KT1=-0.11            KT1L=0               KT2=0.022
UA1=4.31E-9          UB1=-7.61E-18        UC1=-5.6E-11
AT=3.3E4             WL=0                 WLN=1
WW=0                 WWN=1                WWL=0
LL=0                 LLN=1                LW=0
LWN=1                LWL=0                CAPMOD=2
XPART=0.5            CGDO=3.88E-10        CGSO=3.88E-10
CGBO=1E-10           CJ=1.174219E-3       PB=0.824378
MJ=0.4102315         CJSW=1.318145E-10    PBSW=0.8340772
MJSW=0.1             CJSWG=4.22E-10       PBSWG=0.8340772
MJSWG=0.1            CF=0                 PVTH0=5.093002E-5
PRDSW=56.1585333     PK2=2.616923E-3      WKETA=0.0353839
LKETA=0.0163711      PU0=-1.0463987       PUA=-5.37814E-11
PUB=1.543904         PVSAT=-50            PETA0=1E-4
PKETA=-4.815139E-3 )
```

Figure 33. PMOS transistor SPICE model used for the IBM 0.13μm process.

```
.MODEL tsmc18dN NMOS( LEVEL=11
VERSION=3.1          TNOM=27              TOX=4.1E-9
XJ=1E-7              NCH=2.3549E17        VTH0=0.3680311
K1=0.5912273         K2=2.289277E-3       K3=1E-3
K3B=2.5435988        W0=1E-7              NLX=1.738916E-7
DVT0W=0              DVT1W=0              DVT2W=0
DVT0=1.9209561       DVT1=0.4429618       DVT2=-0.0147589
U0=272.9837819       UA=-1.132212E-9      UB=1.883546E-18
UC=2.04883E-11       VSAT=9.701073E4      A0=1.8787969
AGS=0.3794           B0=-1.850639E-8      B1=-1E-7
KETA=-4.764457E-3    A1=0                 A2=1
RDSW=130.862057      PRWG=0.5             PRWB=-0.2
WR=1                 WINT=4.679815E-9     LINT=6.412758E-9
XL=-2E-8             XW=-1E-8             DWG=5.911695E-9
DWB=1.005248E-8      VOFF=-0.0861327      NFACTOR=2.1794075
CIT=0                CDSC=2.4E-4          CDSCD=0
CDSCB=0              ETA0=5.643736E-4     ETAB=-1.416818E-3
DSUB=0.0510645       PCLM=1.7575212       PDIBLC1=0.6724008
PDIBLC2=5.221653E-3  PDIBLCB=-0.1         DROUT=0.8603216
PSCBE1=4.803723E10   PSCBE2=4.871276E-8   PVAG=1.0498691
DELTA=0.01           RSH=6.7              MOBMOD=1
PRT=0                UTE=-1.5             KT1=-0.11
KT1L=0               KT2=0.022            UA1=4.31E-9
UB1=-7.61E-18        UC1=-5.6E-11         AT=3.3E4
WL=0                 WLN=1                WW=0
WWN=1                WWL=0                LL=0
LLN=1                LW=0                 LWN=1
LWL=0                CAPMOD=2             XPART=0.5
CGDO=6.9E-10         CGSO=6.9E-10         CGBO=1E-12
CJ=1.012015E-3       PB=0.7319956         MJ=0.3612578
CJSW=2.171766E-10    PBSW=0.6328732       MJSW=0.1191771
CJSWG=3.3E-10        PBSWG=0.6328732      MJSWG=0.1191771
CF=0                 PVTH0=-3.166914E-3   PRDSW=-1.7313214
PK2=1.550245E-3      WKETA=-1.876456E-3   LKETA=2.206463E-3
PU0=-2.6252751       PUA=-3.86698E-11     PUB=0
PVSAT=1.081788E3     PETA0=1E-4           PKETA=1.035013E-3 )
```

Figure 34. NMOS transistor SPICE model used for the TSMC 0.18μm process.

```
.MODEL tsmc18dP PMOS ( LEVEL=11
VERSION=3.1          TNOM=27              TOX=4.1E-9
XJ=1E-7              NCH=4.1589E17        VTH0=-0.4149264
K1=0.599378          K2=0.0233531         K3=0
K3B=13.5464133       W0=1E-6              NLX=7.089213E-8
DVT0W=0              DVT1W=0              DVT2W=0
DVT0=0.4369111       DVT1=0.2835489       DVT2=0.0831221
U0=120.9245479       UA=1.722692E-9       UB=1.460812E-21
UC=-9.3774E-11       VSAT=2E5             A0=1.5766852
AGS=0.3767784        B1=5E-6              B0=1.019319E-6
B1=5E-6              KETA=0.0186763       A1=0.4410437
RDSW=244.3589747     A2=0.3002749         RDSW=244.3589747
PRWG=0.5             PRWB=-0.5            WR=1
WINT=3.424328E-9     LINT=2.038046E-8     XL=-2E-8
XW=-1E-8             DWG=-1.520363E-8     DWB=-1.879906E-8
VOFF=-0.1033304      NFACTOR=1.796676     CIT=0
CDSC=2.4E-4          CDSCB=0              CDSCD=0
CDSCB=0              ETA0=0.0283932       ETAB=-0.0938944
DSUB=0.7599844       PCLM=1.5542195       PDIBLC1=1.304291E-3
PDIBLC2=0.0163173    PDIBLCB=-1E-3        DROUT=0
PSCBE1=1.842459E9    PSCBE2=5.321454E-10   PVAG=6.3532208
DELTA=0.01           RSH=7.5              MOBMOD=1
PRT=0                UTE=-1.5             KT1=-0.11
KT1L=0               KT2=0.022            UA1=4.31E-9
UB1=-7.61E-18        UC1=-5.6E-11         AT=3.3E4
WL=0                 WLN=1                WW=0
WWN=1                WWL=0                LL=0
LLN=1                LW=0                 LWN=1
LWL=0                CAPMOD=2             XPART=0.5
CGDO=6.86E-10        CGSO=6.86E-10        CGBO=1E-12
CJ=1.135715E-3       PB=0.8647945         MJ=0.4227781
CJSW=1.878779E-10    PBSW=0.6209079       MJSW=0.2821903
CJSWG=4.22E-10       PBSWG=0.6209079      MJSWG=0.2821903
PRDSW=7.5737967      CF=0                 PVTH0=8.450209E-4
PRDSW=7.5737967      PK2=1.878009E-3      WKETA=2.478813E-3
LKETA=-1.186131E-3   PU0=-1.7399784       PUA=-7.09196E-11
PUB=1E-21            PVSAT=50             PETA0=1E-4
PKETA=3.72283E-3 )
```

Figure 35. PMOS transistor SPICE model used for the TSMC 0.18μm process.

```
.MODEL ami06N NMOS ( LEVEL=11
VERSION=3.1          TNOM=27              TOX=1.41E-8
XJ=1.5E-7            NCH=1.7E17           VTH0=0.7086
K1=0.8354582         K2=-0.088431         K3=41.4403818
K3B=-14              W0=6.480766E-7       NLX=1E-10
DVT0W=0              DVT1W=5.3E6          DVT2W=-0.032
DVT0=3.6139113       DVT1=0.3795745       DVT2=-0.1399976
U0=533.6953445       UA=7.558023E-10      UB=1.181167E-18
UC=2.582756E-11      VSAT=1.300981E5      A0=0.5292985
AGS=0.1463715        B0=1.283336E-6       B1=1.408099E-6
KETA=-0.0173166      A1=0                 A2=1
RDSW=2.268366E3      PRWG=-1E-3           PRWB=6.320549E-5
WR=1                 WINT=2.043512E-7     LINT=3.034496E-8
XL=0                 XW=0                 DWG=-1.446149E-8
DWB=2.077539E-8      VOFF=-0.1137226      NFACTOR=1.2880596
CIT=0                CDSC=1.506004E-4     CDSCD=0
CDSCB=0              ETA0=3.815372E-4     ETAB=-1.029178E-3
DSUB=2.173055E-4     PCLM=0.6171774       PDIBLC1=0.185986
PDIBLC2=3.473187E-3  PDIBLCB=-1E-3        DROUT=0.4037723
PSCBE1=5.998012E9    PSCBE2=3.788068E-8   PVAG=0.012927
DELTA=0.01           MOBMOD=1             PRT=0
UTE=-1.5             KT1=-0.11            KT1L=0
KT2=0.022            UA1=4.31E-9          UB1=-7.61E-18
UC1=-5.6E-11         AT=3.3E4             WL=0
WLN=1                WW=0                 WWN=1
WWL=0                LL=0                 LLN=1
LW=0                 LWN=1                LWL=0
CAPMOD=2             XPART=0.4            CGDO=1.99E-10
CGSO=1.99E-10        CGBO=0               CJ=4.233802E-4
PB=0.9899238         MJ=0.4495859         CJSW=3.825632E-10
PBSW=0.1082556       MJSW=0.1083618       PVTH0=0.0212852
PRDSW=-16.1546703    PK2=0.0253069        WKETA=0.0188633
LKETA=0.0204965)
```

Figure 36. NMOS transistor SPICE model used for the AMI 0.6μm process.

```
.MODEL ami06P PMOS ( LEVEL=11
VERSION=3.1          TNOM=27              TOX=1.41E-8
XJ=1.5E-7            NCH=1.7E17           VTH0=-0.9179952
K1=0.5575604         K2=0.010265          K3=14.0655075
K3B=-2.3032921       W0=1.147829E-6       NLX=1.114768E-10
DVT0W=0              DVT1W=5.3E6          DVT2W=-0.032
DVT0=2.2896412       DVT1=0.5213085       DVT2=-0.1337987
U0=202.4540953       UA=2.290194E-9       UB=9.779742E-19
UC=-3.69771E-11      VSAT=1.307891E5      A0=0.8356881
AGS=0.1568774        B0=2.365956E-6       B1=5E-6
KETA=-5.769328E-3    A1=0                 A2=1
RDSW=2.746814E3      PRWG=2.34865E-3      PRWB=0.0172298
WR=1                 WINT=2.586255E-7     LINT=7.205014E-8
XL=0                 XW=0                 DWG=-2.133054E-8
DWB=9.857534E-9      VOFF=-0.0837499      NFACTOR=1.2415529
CIT=0                CDSC=4.363744E-4     CDSCD=0
CDSCB=0              ETA0=0.11276         ETAB=-2.9484E-3
DSUB=0.3389402       PCLM=4.9847806       PDIBLC1=2.481735E-5
PDIBLC2=0.01         PDIBLCB=0            DROUT=0.9975107
PSCBE1=3.497872E9    PSCBE2=4.974352E-9   PVAG=10.9914549
DELTA=0.01           MOBMOD=1             PRT=0
UTE=-1.5             KT1=-0.11            KT1L=0
KT2=0.022            UA1=4.31E-9          UB1=-7.61E-18
UC1=-5.6E-11         AT=3.3E4             WL=0
WLN=1                WW=0                 WWN=1
WWL=0                LL=0                 LLN=1
LW=0                 LWN=1                LWL=0
CAPMOD=2             XPART=0.4            CGDO=2.4E-10
CGSO=2.4E-10         CGBO=0               CJ=7.273568E-4
PB=0.9665597         MJ=0.4959837         CJSW=3.114708E-10
PBSW=0.99            MJSW=0.2653654       PVTH0=9.420541E-3
PRDSW=-231.2571566   PK2=1.396684E-3      WKETA=1.862966E-3
LKETA=5.728589E-3)
```

Figure 37. PMOS transistor SPICE model used for the AMI 0.6μm process.

Appendix B: Results Data

The results of the verification of the calibration method are presented here. Table 5 shows the results of the IBM 0.13μm process. Table 6 shows the results of the TSMC 0.18μm process. Table 7 shows the results of the AMI 0.6μm process. All of the gates were tested under two different load capacitances, FO1 and FO4, and three different input rise and fall times. t_{MIN} in the tables is the minimum propagation delay that yields a 20% error between the simulated delay and the targeted delay. t_{MAX} is the maximum propagation delay that yields 20% error. Also, the two right columns in the tables show the error percentage at the minimum transistor width and the "maximum" transistor width, which is taken to be fifty times the minimum transistor length. These errors are approximately the largest errors produced by the calibration method.

Table 5. Verification Results for the IBM 0.13μm process.

IBM 0.13μm						
Gate	Load	t_{IN} (ps)	Propagation Delay Range (ps)		% Error in Propagation Delay	
			t_{MIN}	t_{MAX}	$W=W_{MIN}$	$W=L_{MIN}*50$
INV	FO1	34	18	39.9	-9.3	27
		222	Underestimates Size		21.5	60.2
		410	55	69.2	9	34.1
	FO4	34	20.3	55	-26.2	9.9
		222	55	130	-22.6	43.9
		410	55	150	-25.4	24.3
NAND2	FO1	34	26.3	64.4	-12.5	16.3
		222	65	83	5.7	48.7
		410	70	103	2.6	27.8
	FO4	34	31.8	100	-28.9	9.1
		222	65	140	-32.6	33.7
		410	67	150	-33.5	15.4
NAND3	FO1	34	40.4	95.5	-15.7	8.7
		222	70	115	-8.7	27.8
		410	77	135	-10	14.5
	FO4	34	45.9	140	-27.5	5.4
		222	66.8	150	-36.8	18.4
		410	89.6	160	-36.4	10.4
NAND4	FO1	34	57.1	132	-19.5	2.1
		222	79.5	151	-20.7	9.9
		410	101	169	-17.4	12.9
	FO4	34	62.5	200	-29.8	-0.7
		222	88.8	150	-41.5	7.7
		410	113	170	-40.3	10.4
NOR2	FO1	34	45	60.6	7.3	28.8
		222	77	77	20.3	58.9
		410	94.5	94.5	19.5	19.5
	FO4	34	43	134	-12.7	19.1
		222	80	171	-13.9	41.3
		410	100	208	-16.4	41.2
NOR3	FO1	34	80	92	13.4	25.1
		222	95	109	14.2	35.5
		410	123	123	20.2	41.4
	FO4	34	77.7	165	-11.5	18.9
		222	100	203	-11.8	24.7
		410	120	237	-15.5	30.4
NOR4	FO1	34	115	134	16.1	22.1
		222	115	150	7.5	14.6
		410	Underestimates Size		21.9	29.6
	FO4	34	123	207	-12.9	18.3
		222	132	244	-11	9.3
		410	150	273	-18.2	24.4

Table 6. Verification Results for the TSMC 0.18μm process.

TSMC 0.18μm						
Gate	**Load**	**t_{IN} (ps)**	**Propagation Delay Range (ps)**		**% Error in Propagation Delay**	
			t_{MIN}	**t_{MAX}**	**$W=W_{MIN}$**	**$W=L_{MIN}*50$**
INV	**FO1**	**40**	21	43	-29.4	23.3
		320	40	74	-25	37.9
		600	51.3	90	-35	18.1
	FO4	**40**	25.3	70	-40.8	13
		320	55	120	-46.5	26.2
		600	68.8	140	-48.8	9.3
NAND2	**FO1**	**40**	33.7	69	-30.3	20.5
		320	56	100	-31.6	13.6
		600	83.7	120	-38.1	5.9
	FO4	**40**	39.8	125	-38.5	15.6
		320	70	150	-44.3	21.2
		600	101	160	-49.2	-5
NAND3	**FO1**	**40**	52	95	-33.5	14.2
		320	81.9	120	-36.3	15.6
		600	118	150	-40.2	-8.1
	FO4	**40**	58	170	-39.5	11.6
		320	93	180	-44.1	8.8
		600	135	160	-47.6	-13.1
NAND4	**FO1**	**40**	73.7	130	-35.5	8.7
		320	112	160	-38.1	1.8
		600	155	170	-42.1	-16.4
	FO4	**40**	79.8	220	-40.2	7.3
		320	123	180	-47.4	-2.9
		600	172	172	-30.3	-20
NOR2	**FO1**	**40**	47	73	-23.1	25
		320	80	114	-20.4	36.1
		600	94	140	-29.4	25.2
	FO4	**40**	54	120	-35	17.6
		320	90	170	-40.8	23
		600	106	175	-47	14.3
NOR3	**FO1**	**40**	83.5	110	-23	18
		320	113	155	-24.8	-7.6
		600	133	180	-32.8	-12.6
	FO4	**40**	98	170	-31	13.9
		320	133	200	-41.8	-9.8
		600	155	230	-47.3	-15
NOR4	**FO1**	**40**	137	163	-24.7	-11.9
		320	177	200	-34.2	-16.7
		600	195	195	-38.7	-19.2
	FO4	**40**	156	230	-28.9	-11.8
		320	202	230	-44.3	-17.7
		600	221	221	-49.5	-20

Table 7. Verification Results for the AMI 0.6μm process.

AMI 0.6μm						
Gate	Load	t_{IN} (ps)	Propagation Delay Range (ps)		% Error in Propagation Delay	
			t_{MIN}	t_{MAX}	$W=W_{MIN}$	$W=L_{MIN}*50$
INV	FO1	125	48	130	16.9	19.4
		963	135	140	32.5	51.7
		1800	122	200	31.4	15.2
	FO4	125	70	375	-14.5	3.7
		963	150	400	-28.2	33.4
		1800	185	320	-41.7	-8.2
NAND2	FO1	125	89	213	20.8	6.4
		963	150	302	13.2	31.1
		1800	210	270	-33.1	-14.7
	FO4	125	112	543	15.2	-7.4
		963	180	440	-31.4	-20.5
		1800	270	330	-44.8	-18.7
NAND3	FO1	125	140	324	16.9	-9.6
		963	300	428	-14.3	-32.7
		1800	Overestimates Size		-36.4	-28.9
	FO4	125	164	763	14.7	-10.1
		963	Overestimates Size		-34.7	-31.8
		1800	Overestimates Size		-47.2	-29.6
NAND4	FO1	125	203	419	15.4	-11.9
		963	470	587	-20.2	-38.7
		1800	Overestimates Size		-38.5	-40.4
	FO4	125	230	897	7.1	-12.8
		963	Overestimates Size		-36.1	-37.7
		1800	Overestimates Size		-49.6	-39.8
NOR2	FO1	125	118	155	34.4	13.2
		963	175	190	-54.5	18.6
		1800	230	280	-33.7	-10.5
	FO4	125	157	448	16.3	-9
		963	234	330	-31.3	-11.6
		1800	300	350	-41	-15.1
NOR3	FO1	125	227	240	44.7	-15.1
		963	Overestimates Size		-32.2	-28.1
		1800	Overestimates Size		-40.9	-35.4
	FO4	125	284	480	28	-15.1
		963	Overestimates Size		-34	-28.9
		1800	Overestimates Size		-42.1	-31
NOR4	FO1	125	Overestimates Size		50.6	-21.4
		963	Overestimates Size		-37.5	-37.7
		1800	Overestimates Size		-45.4	-40.6
	FO4	125	440	440	37.3	-20
		963	Overestimates Size		-36.7	-37
		1800	Overestimates Size		-59.4	-40.1